

CORE
BOOKS

実用インターフェース設計法

マイコン活用のためのハードウェア技術入門

畔津明仁 著



- 良い製品には良い設計思想が活かされており、良い設計思想には、必ずしっかりとしたバックグラウンドがあります。
- ますます IC/LSI 化が進み高度化するエレクトロニクス技術も、バックグラウンドとなる技術さえあれば怖くありません。
- CORE BOOKS は、あなたのエレクトロニクス技術のバックグラウンドづくりを応援する CQ 出版社の新しい書籍シリーズです。

〈カバー〉

- デザイン／宰 良二
- フォト／佐瀬 真

実用インターフェース設計法

マイコン活用のためのハードウェア技術入門

畔津明仁 著

CQ出版社

まえがき

わたしたちが少し複雑な機能を持つ装置を設計するとき、その中にマイクロプロセッサを組み入れることは現在では当たり前になりました。マイクロプロセッサ（またはそれを含む“マイコン”）とは、それほど便利で汎用性のある部品です。しかし、その便利さを発揮するためには、組み込まれる装置ごとにインターフェースを設計しなければなりません。

一方、市販のパーソナル・コンピュータの価格が下がり、マイコンは作る時代から買う時代になりました。このため、誰でも簡単にマイコンの恩恵にあずかれるようになりましたが、ちょっとしたインターフェース回路ですら自分では作れない人も増加しています。

とかく、マイコンを使いこなすこととはソフトウェアが作れることであると思われるがちです。しかし、より広範囲の応用を考えるならば、ハードウェアの知識も必要です。

本書は、マイコンのソフトウェアには触れたことがあるがハードウェアには自信がないという初心者の方々に、設計のヒントとしていただこうと意図したものです。また、より上級の方々にも基礎知識の復習となれば、幸いです。

なお、マイコンを使って何かを作り上げることは、きわめて総合的かつ創造的な仕事であり、本書一冊ではとうてい全部をカバーすることはできません。このため、ソフトウェアをはじめとして本書に書かれていない部分は他の書籍等を参考にさせていただくとともに、お気付きの点は遠慮なくご指摘下さい。

1985年 春

著者

目 次

第1章 マイコン・システムの構成法	9
1.1 マイコンを使ったシステムの形態	10
1.2 マイコンを使ったシステムの設計手順	13
必要とされる入出力を見極める(13)/入出力の仕様を定める(14)/ ソフトウェア/ハードウェアの分担を決める(16)/ソフトウェア/ ハードウェアの設計(18)/ソフトウェア/ハードウェアのデバッグ (19)	
コラムA マイコンとは	19
第2章 マイコンの構造と動作	21
2.1 マイコンの動作	21
命令読出し/命令実行(22)/バス構成(23)/バスの動作(23)	
2.2 マイコン・システムにおけるバス	25
バス負荷の考え方(25)/バス・バッファ(26)	
2.3 Z80 CPU バス	27
Z80 のバス構成(28)/バス・サイクルの種類(29)	
2.4 各種8ビットCPUのバス・タイミング	33
68系(6809/6802/6800)のバス(33)/8048系(8048~8051/8031/35/39など) のバス(35)/Z80系(86××)のバス(37)/8088(ミニマム・モード), 8085, NSC800のバス(39)	
コラムB バスの種類	41
コラムC バス・マスタとバス・スレーブ	42
コラムD スリーステート	43
コラムE バス・タイミングの記述方法	44
第3章 マイコン入力インターフェースの実際	47
3.1 入力信号の種類と処理法	47
入力信号の形態とレベル(47)/入力信号の速度(48)/入力信号の数と処 理方法(50)	
3.2 入力インターフェースのマイコン側の設計	51
マイコンのデータ・バスを駆動するスリーステート(3ステート)・バッ ファ(52)/アドレスや制御信号が所定の組合せとなったことを検出する	

デコーダ(52)

例3-1	基本的な入力インターフェース	52
例3-2	ビット単位の入力方法	56
例3-3	多数の信号を入力するインターフェース	58
3.3	入力インターフェースの外部機器側の設計	60
例3-4	基本的なスイッチの入力方法——チャタリング対策①	61
例3-5	スイッチのチャタリング対策②	65
例3-6	多数のスイッチの入力①——エンコード	66
例3-7	多数のスイッチの入力②——ダイナミック・スキャン	68
例3-8	オープン・コレクタ信号の入力	71
例3-9	他のロジック・レベルの入力	74
例3-10	状態を記憶する	76
例3-11	ストローブ付きの入力	76
例3-12	ハンドシェイク入力	79
例3-13	基本的なアナログ入力①——レベル検出	81
例3-14	基本的なアナログ入力②——8ビット中速 A-D 変換器	83
例3-15	高速のアナログ入力——高速 A-D 変換上の問題点	85
例3-16	高分解能アナログ入力——必要な精度と分解能を得る方法	87
コラム F	デコーダのいろいろ	91
コラム G	フル・デコードとリニア・デコード	91
コラム H	スイッチの種類について	93

第4章 マイコン出力インターフェースの設計 97

4.1 出力信号の種類と処理方法 98

出力信号の形態とレベル(98)/出力信号の速度(99)/出力信号の同時性
(99)/出力信号の数(100)

4.2 出力インターフェースのマイコン側の設計 101

例4-1	基本的な出力インターフェース	102
例4-2	ビット単位でオン/オフできる出力インターフェース	105
例4-3	多数の信号を出力するインターフェース	106
例4-4	多数の信号の同時出力	108
例4-5	CMOS 出力インターフェース	111
例4-6	パルス信号の出力	113
例4-7	ストローブ付き出力とセントロニクス型ハンドシェイク出力	115

4.3 出力インターフェースの外部機器側の設計	118
例4-8 ランプ(白熱電球)の点灯	118
例4-9 LEDの駆動	120
例4-10 液晶表示器の駆動	124
例4-11 リレーの駆動	127
例4-12 アナログ信号の出力①——アナログ・スイッチによる方法	131
例4-13 アナログ信号の出力②——D-A変換器による方法	134
コラムI Dフリップフロップとトランスパレント・ラッチ	138
コラムJ セントロニクス型インターフェース	140
コラムK CPUを選ばない万能I/Fは可能か①——8ビット汎用CPU	142
コラムL CPUを選ばない万能I/Fは可能か②——MOTEL	144
コラムM R-2RラダーとD-A変換器	146
第5章 高度な入出力インターフェースの設計	149
5.1 直列通信回線のインターフェース	151
例5-1 直列通信回線	151
5.2 映像信号のインターフェース	154
例5-2 映像信号の出力	154
例5-3 映像信号の入力	158
5.3 分散処理のインターフェース	161
例5-4 分散処理(複数CPUシステム)	161
第6章 入出力インターフェース・ユニットの実際	165
全体の構成(166)	
6.1 CPU基板	166
CPU基板の特徴(167)/CPUの選定(167)/メモリ(167)/直列通信回線 (RS-232C インターフェース)(168)/並列インターフェース(170)/その 他(170)	
6.2 多数のスイッチの入力インターフェース	170
ダイナミック・スキャン(174)/動作の詳細(176)	
6.3 フォト・カプラで絶縁された汎用入出力インターフェース	178
アドレス・デコーダ(179)/入出力の極性の変更など(179)/入出力の本 数と汎用性(180)	
6.4 ケーブルの延長を考慮した入出力インターフェース	182

6.5 分散処理のための直流モータ制御用基板	183
直流モータ制御に必要な入出力(184)／CPUの選定(185)／ホストとの間のデータ交換(186)／データ交換のタイミング(188)／その他の入出力(188)	
Appendix マイコン・システムのデバッグの例(Z80用モニタ・プログラム)	191
I. バグをなくすために	191
設計の注意点(191)／製作上の注意点(191)	
II. Z80用モニタ・プログラム	192
III. デバッグの実際	194
ハードウェアのデバッグ(194)／モニタ・プログラムによる各部の検査(197)	
〈リスト〉 Z80用モニタ・プログラム	198
参考文献	207
索引	208

巻末綴込付録

- 付図1 CPU ボード
- 付図2 フォト・カプラで絶縁された入出力インターフェース・ボード
- 付図3 操作箱用インターフェース・ボード
- 付図4 モータ制御用ボード

編集担当／山本 潔

〈カバー〉

デザイン／幸 良二

フ ォ ト／佐瀬 真

第1章

マイコン・システムの構成法

マイコンの応用範囲はますます広がりつつあります。以前はマイコンそれ自体が完結したシステムと考えられていたのに対して、最近の使われ方を見ると、マイコンは汎用コントローラとして大きなシステムの一部に組み入れられることが多くなりました。

しかも、マイコンの守備圏はきわめて広く、システムの分散処理指向とあいまって、多品種少量生産品のコントローラとしては欠かせないものとなっています。

もちろん、これはマイコンの汎用性——ソフトウェアと一部分のハードウェアとを取り換えれば“何でも”できる点——を買われてのことです。しかし、その利点を活かすためには、似てはいても少しずつ異なるソフトウェアとハードウェアとを、そのシステムごとに設計しなければなりません。

このわずらわしさは、とくにソフトウェアについて、最近やかましくいわれるようになってきました。ハードウェアに関しては従来はさほど重視とされていませんでしたが、メーカ製のマイコンの価格が量産効果によって驚くほど下がり、その中身の詳細を知らずに使う時代となってきた結果、逆に新たな問題となっているようです。つまり、ごく簡単なインターフェースを製作するにも、必要な情報が公開されていなかったり、担当者が細部にわたる知識をもっていなかったりするという理由で最適な設計が行えない事態が少なからず存在するのです。

最適でなくとも確実に動作すれば良いのですが、しばしば頭を痛めるのは、既製のマイコンと外部との■で簡単なやりとり（たとえばスイッチを接続したり、ランプを点灯したりという程度の）を行うためのインターフェースを追加したところ、“ときどき”マイコンがエラーを起こす、という症状がでることです。

もちろん、追加したインターフェース回路はちゃんとした書籍や雑誌に発表されたもの

であり、単体で機能チェックを行うと、完全に動作しています。マイコンのほうも単体ではエラーなど生じません。では、このような事態がなぜ生じるのか、また正しく設計するにはどうすればよいのか、それらの点を説明することを本書の目的とします。

1.1 マイコンを使ったシステムの形態

マイコンといっても、1個のIC（シングルチップ・コンピュータ）に電源を接続しただけのものもあれば、CRT（ブラウン管）やキーボードのついたパーソナル・コンピュータもあります。また、その用途もいろいろです。

図1.1に示すのは、マイコンを使った種々のシステム例です。同図(a)は1枚の基板上にCPUやインターフェース回路を作り、組み込み型のコントローラとしたものです。以前なら

図1.1 (a) マイコン・システムの形態例(専用の制御基板)

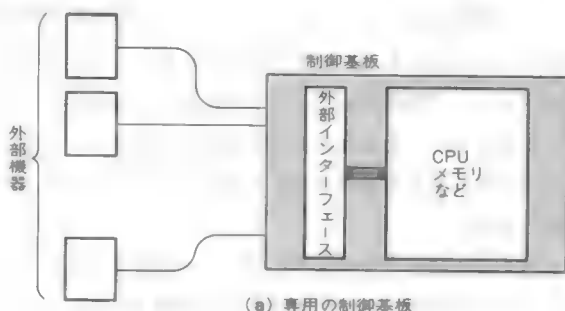
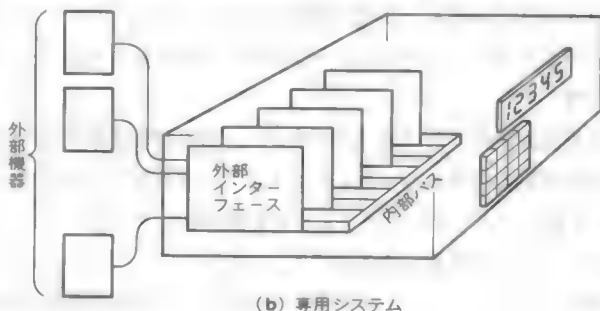


図1.1 (b) マイコン・システムの形態例(専用システム)



ばハードウェアで組んでいた制御回路も、最近はこのようにマイコンを使ったものが増えました。このようなシステムでは、仕様の範囲を正確に見定めて、必要な機能をコンパクトにまとめることが重要です。

図1.1(b)に示すのはやや規模が大きい場合で、マイコンを使ったコントローラを別の箱に入れて各種の制御を行わせようとするものです。このような構成では、回路規模を小さ

図1.1 (c) マイコン・システムの形態例(パーソナル・コンピュータを用いた構成)

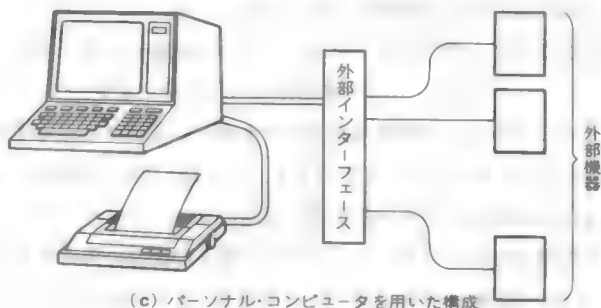
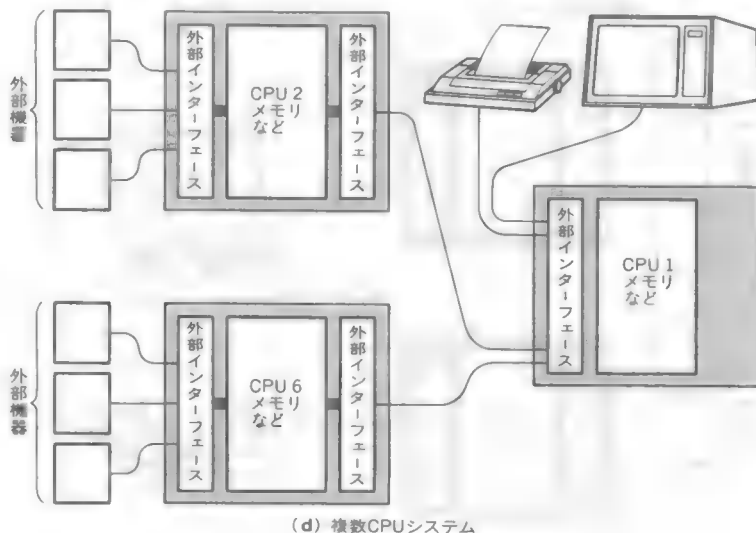


図1.1 (d) マイコン・システムの形態例(複数CPUシステム)

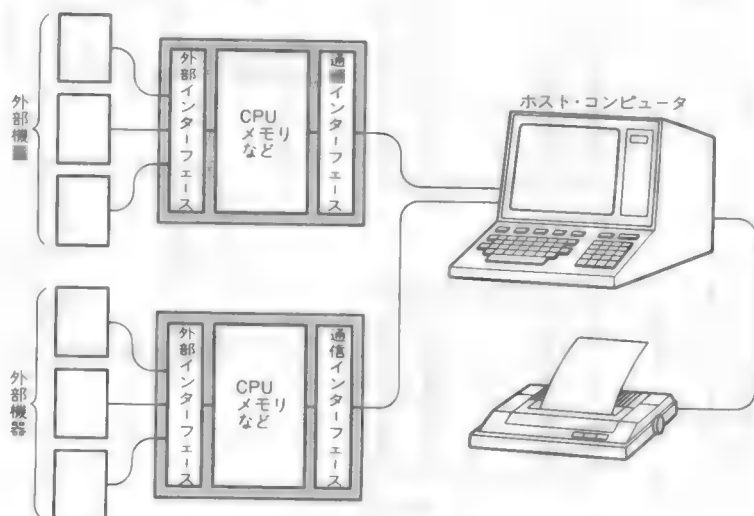


くすることよりも、専用コントローラとしての機能を充実させることが重要です。

図1.1(c)は市販のパーソナル・コンピュータにインターフェース部分を増設したものです。パーソナル・コンピュータにはCPUなどマイコンとしての中核部分のほか、CRT表示器やキーボードなどの入出力装置、直列通信回線(RS-232Cなど)やプリンタ・インターフェースなど、汎用的な入出力インターフェースがすでに付属しています。したがって、これらの機能の一部を使うつもりならば、安価で便利なシステムとなります。ただし、ごくかぎられた用途を除いて、入出力インターフェース回路を追加する必要があります。その場合、アドレスの割当てやバス駆動能力などの点で、パーソナル・コンピュータから制約を受けることになります。このためインターフェース回路を自作するにしろ、市販の基板を使うにしろ、パーソナル・コンピュータ本体についてハードウェア/ソフトウェア両面の十分な資料を入手してから、利用が可能か否かを検討しなければなりません。

図1.1(d)は複数のCPUを使ったシステムです。これは一種の分散処理であり、ホストCPU(CPU 1)は全体に関係するデータ処理や制御を行い、スレーブCPU(CPU 2~6)は個々に接続された機器とのインターフェースや、個々の機器に限定されたデータ処理を行います。今日、CPUの価格は非常に安くなっており、このようにインターフェース自体

図1.1(e) マイコン・システムの形態例(パーソナル・コンピュータを中心とした分散処理システム)



(e) パーソナル・コンピュータを中心とした分散処理システム

をマイコンで作ることも増えてきました。このような構成では処理速度も上がり、ソフトウェアも別々に作れば良いので、むしろ簡単になるでしょう。また、同図(8)のように中心にホスト・コンピュータ(パーソナル・コンピュータ)を置けば、パーソナル・コンピュータと専用機との長所を補い合ったシステムが作れます。このようなシステムでは、各CPU間の仕事の分担のしかたと、CPU間の連絡方法(やはりインターフェース)とが設計のポイントとなります。

1.2 マイコンを使ったシステムの設計手順

以上のように、マイコンを使ったシステムには様々な形態があります。また、接続したい対象も非常に多く、いざ設計しようとしても何から手を付けたら良いかわからないこともあると思います。

そこで、設計の■に最小限必要なステップとして■1.1のような順序を考えてみます。もちろん、設計はこの順序で行わなければならないものではありませんが、このようなシーケンスが頭に入っていれば設計に必ず役立ちます。

■1.1 入出力を見極める

まず、システム全体を頭に入れて、どのような入出力が必要かを考えます。入力とは■気信号で外部から与えられるもののほか、各種のセンサやトランスジューサのこともあります。また、人間が操作するものとそうでないものがあります。

一方、出力も電氣的なものだけではなく、音や光による合図、プリンタやCRTへの表示なども忘れないように拾い上げていきます。

一般に、マイコンに接続されるものの1例を図1.2に示します。結論からいえば、マイ

■1.1 入出力インターフェース設計の手順の例

- (1) どのような入出力が必要か。
- (2) 各入出力について、どのような機能(仕様)が必要か
 - 使用目的と使用方法。
 - 形態やレベル
 - 雑音対策、伝達距離、インピーダンス
 - マイコン側から見た速度
- (3) ソフトウェア/ハードウェアの分担を決める。
- (4) ソフトウェア/ハードウェアの設計
- (5) デバッグ

コンには“何でも接続したい”という要求があり、この図に示したものも1例に過ぎません。むしろ、この図に出ていないものを発見し、独創的なインターフェースを設計してマイコンに接続することこそ、本当にマイコンを活用することにつながります。

●入出力の仕様を定める

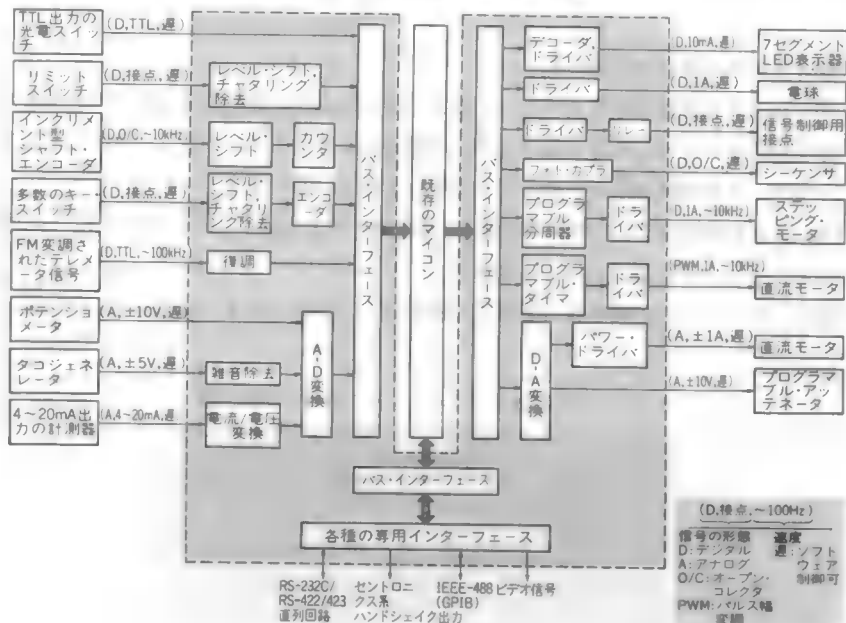
つぎに、それぞれの入出力について具体的な内容を決めます。この作業はきわめて重要で、インターフェース回路の設計の可否はこの段階で左右されます。

では、入出力の仕様として何に注目すれば良いのでしょうか。

(1) 信号の使用目的と使用方法 まず何のために必要な信号かを良く考えて下さい。後で良く考えたら必要な信号ではなかったとか、逆に信号が不足したというのは、この段階での失敗です。

次に重要なのは信号の使用方法です。たとえば、人間（操作者）が操作するスタート・スイッチにしても、

図 1.2 マイコンに何をつなぐか



- (i) 押しボタン・スイッチ（モーメンタリ）にして1回押せばスタートする。
- (ii) 押しボタン・スイッチ（モーメンタリ）だが押している間だけ動く。
- (iii) トグル・スイッチにしてパチンと切り換えてしまう。

など。いろいろな操作方法があります。もっとも操作しやすく、間違いが少ないものを■ぶのが重要で、このことがマン・マシン・インターフェースの第一歩につながります。

一方、機械と機械との信号のやりとりにも同じことがいえます。たとえば動作中か否かを別の機器に知らせる電気信号でも、動作中であることをレベルで示すのと、動作終了時にパルスを出すのとでは、それぞれ長所、短所があります（詳細は■3章を参照）。

(2) 信号の形態やレベル つぎに可能なものについては電氣的な仕様を決めていきます。信号の形態やレベルとは、それがアナログかデジタルか、接点か電圧か、その電圧はどれくらいか、などを指します。

(3) 雑音、信号の伝達距離■、インピーダンス 外部機器側の信号として高電圧や大電流を扱う場合、これがマイコン本体や他のインターフェースに及ぼす■■を考慮します。また逆に、微小信号を扱う場合は、ほかからの雑音混入に注意します。特に、マイコン自体は大きなデジタル回路であり、周囲に少なからぬ雑音を与えることがあります。

一方、長い信号線が必要とする場合や、信号源インピーダンスが高い場合、信号伝達に注意が必要となります。

(4) 必要とされる速度 これは、ハードウェアとソフトウェアとの分担を決める上で非常に重要ですから、慎重に検討します。

速度には、つぎの2種類があります。

(i) 変化の周期

(ii) 必要とされるタイミングとのズレ

変化の周期は入出力のサンプリング間隔を選ぶのに重要です。ソフトウェアでループを作った場合、それを1回まわるには簡単なものでも■10～100 μs が必要です。したがって、入力であっても出力であっても、ソフトウェアを用いて100 μs 以下の信号変化周期を扱うのは困難です。

つぎに、必要とされるタイミングとのズレについて考えます。たとえば、変化周期は遅いものの、ストロブ信号が印加されてから一定時間以内にデータを入力、あるいは出力しなければならないことがあります。また、複数の入力信号が同時に変化したか否かを■べたり、複数の出力信号を同時に変化させたい場合もあります。

ソフトウェアは必ずしも“柔軟”ではありません。たとえば別々のアドレスに割り当て

られた信号を同時に変化させることは、ハードウェアで手段を設けていないかぎり不可能です。したがって、この問題はハード/ソフトのトレード・オフとも関連して検討しておく必要があります。

●ソフトウェア/ハードウェアの分担を決める

すでにおわかりのように、マイコンはソフトウェアとハードウェアとの適切な組合せがあって初めて所定の動作が可能となります。

しかし、その場合、ソフトとハードとの分担を決めることが必要となります。これは仕事の種類によってつぎの3種類に分けられます(図1.3)。

- (1) ハードウェアでは可能だが、ソフトウェアでは不可能またはきわめて困難なもの。
- (2) ハードウェア、ソフトウェアのいずれによっても可能なもの。
- (3) ソフトウェアでは可能だが、ハードウェアではきわめて困難なもの。

まず(1)に属する仕事に対しては必ずハードウェアを設けなければなりませんので、この範囲を定めることはきわめて重要です。

(1) ハードウェアでは可能だが、ソフトウェアでは困難なもの まず、仕事の内容を2つに分けて考えてみます。その第1は信号の形態や、他の電気的仕様の変換です。これは、ほとんどの場合にハードウェアの仕事になるでしょう。たとえば、CPUに直接アナログ信号を接続しても、ソフトウェアではどうしようもありません(図1.4)。

第2の仕事は、データの転送や変換などの処理です。これについては現在のマイコンと、それに対する要求とから見て、ソフトウェアでもっとも苦手なのは■1に高速性を要求さ

図1.3 ハードウェア/ソフトウェアの分担

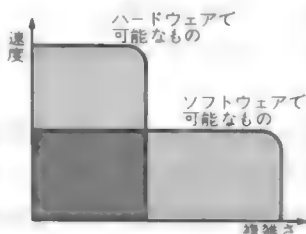
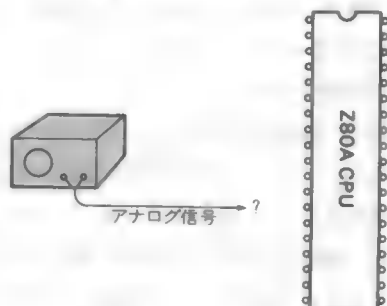


図1.4 デジタル用の端子にアナログ信号はつなげない



れるもの、第2に多数の事象の並列処理であるといえます。

まず、前に述べたように、一定以上の速度が必要とされる場合はソフトウェアでは対応できません。

また、複数の事象の並列処理もソフトウェアでは困難です。簡単なものでは多数の入力信号が“同時”に変化したか否かを調べたり、多数の出力信号を“同時”に変化させることは、ソフトウェアだけでは困難です。これは、現在のマイコンが一連の命令列を順に実行する（いわゆるノイマン型）かぎり、別々のアドレスに割り当てられた信号を厳密に同時に扱うことは不可能だからです。したがって、このような仕事もハードウェアのお世話になります。

(2) ハードウェア/ソフトウェアのいずれでも可能なもの これは“簡単な内容であっても速度も遅くてよいもの”になります。ただし、ハード、ソフトの両方で可能であっても、それぞれに特徴があることを忘れてはなりません。

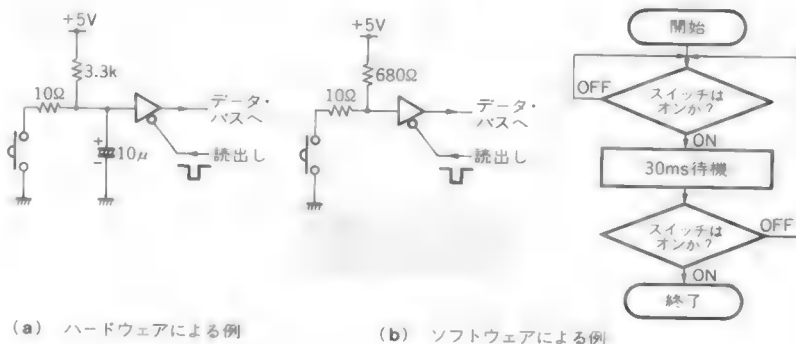
たとえば、スイッチのチャタリング除去（第3章参照）をハード、ソフトのそれぞれで行った場合を考えてみます（図1.5参照）。一般にソフトの利点といわれているのは、

- (i) 部品数が減る
- (ii) 時定数を簡単に（ハンダなしに）変更できる

という点です。このほかに、部品の経時変化の影響を受けにくい点も見逃せません。

しかしながら、図1.5(a)の回路は図1.6のようなパルス性雑音の除去もできますが、ソフトウェアではできないことがあります（たとえば、マイコンの動作自体に同期した雑音は除去できない）。また、実際に稼動し始めたあとのソフトウェアを変更するのは困難な場

図1.5 チャタリング除去



合も多く、上記(ii)は必ずしもソフトの利点とはいえません。

このように、ハード、ソフト両方で可能なものでも両者の特徴をよく整理して選択する必要があります。

(3) **ハードウェアでは困難なもの** ハードウェアで困難な仕事とは、複雑な処理や演算を必要とするもののことです。普通はこのような仕事はソフトで行いますので、本書では触れません。ただし、経済性などを無視すればソフトで実現できるものは原則としてハードに置き換えることが可能（逆は正しくない）だという点は覚えておく必要があります。

●ソフトウェア/ハードウェアの設計

入出力やソフト/ハード分担の仕様が決まったら、いよいよ設計に入ります。このうち、入出力インターフェースのハードウェアについては本書で詳しく説明します。ソフトウェアは他の参考書等をご覧下さい。その場合に注意すべきなのは次のようなことです。

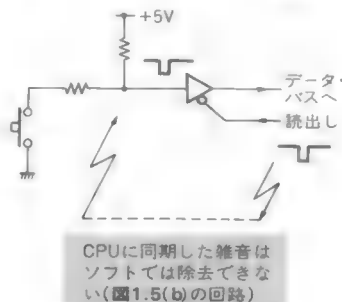
(i) 常にシステム全体への要求を頭に入れておくこと。

仕様の1つ1つを満足することをなおざりにしてはいけません。しかし、それにはシステム全体が働くことが大前提となります。

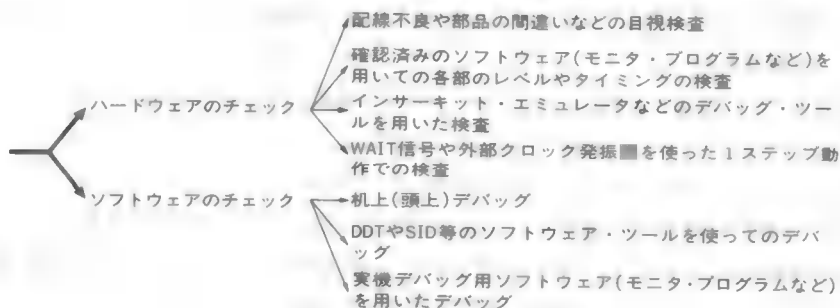
(ii) 入手可能な情報をフルに活用すること。

たとえば市販のパーソナル・コンピュータにインターフェースを追加する場合、少なくともその中身について知っておく必要があります。ソフトウェアを設計するにも、使用するCPUの動作は知っておくべきです。

図 1.6 ソフトウェアでは除去できない雑音



■ 1.2 デバッグの方法——その1例



●ソフトウェア/ハードウェアのデバッグ

マイコンを使ったシステムでもっとも困るのがデバッグです。へたをすると、原因がハードかソフトかわからないまま、全く動かないシステムができてしまいます。

デバッグの方法そのものは本書の目的ではありませんが、デバッグを考慮したインターフェースを設計することは重要です。とくに、組込み型のコントローラでは、最初から実機デバッグを考慮しておかないかぎり、動かなかったりしたときに手のつけようがありません。これには表1.2に示すような方法があるので参考にして下さい。

コラムA マイコンとは

世の中にはマイコン、パソコン、マイクロプロセッサなど似たような名称が多く存在します。この種の単語は、定義など関係ないと無意識の選択によって使われることもあれば、意識的に異なった意味で区別して使われることもあり、事態をより複雑にしています。そこで、本書ではつぎのような意味で使うことを決めておきます。

(1) マイコン (マイクロコンピュータ)

単体で動作し、CPU(マイクロプロセッサ)、メモリ、入出力手段などの基本的な要素を含むもの。たとえば、市販のパーソナル・コンピュータも、組込み用のCPU基板に電源を付けたものも、マイコンの1つである。本書では“マイコン”という語が数多く登場するが、このように広い意味で使用している点に留意してほしい。

(2) CPU, マイクロプロセッサ, MPU

CPU は計算機の中央処理装置 (Central Processing Unit) のことだが、それを LSI

化したものがマイクロプロセッサ。マイコンは通常、マイクロプロセッサを中心として構成される。MPU (Micro Processing Unit) は商標で、マイクロプロセッサと同じ意味である。

ただし、本書ではマイクロプロセッサやその機能を含んだIC (たとえばシングルチップ・コンピュータ) をCPUと記すことがある。

(3) ワンチップ・マイコン、シングルチップ・コンピュータ

1個のICにCPUのほかメモリやI/O機能を含んだもの。電源や少数個の外付け部品を加えると、マイコンと呼べる能力をもつ。典型的なものとしてインテル 8048 系やモステック 3870 系などがある。家電用には各社各様のシングルチップ・コンピュータが多く使われている。

第 2 章

マイコンの構造と動作

マイコンのインターフェースを設計するには、マイコンの構造や動作を知っておく必要があります。

2.1 マイコンの動作

そこで、簡単な例をあげて、実際のコンピュータの動作を調べてみましょう。

図 2.1(a)に示すのは、Z80 CPU を用いた簡単なプログラムです。これは 8000₍₁₆₎ 番地の内容に 30₍₁₆₎ を加えて 8001₍₁₆₎ 番地に格納するもので、8000 番地を入力ポート、8001 番地を出力ポートとすれば、概念的には図 2.1 (b) のようなハードウェアと同じ働きをします。

しかし、図 2.1 (a) と (b) との働きは良く似ていても、実際の動き方は全く異なります。図

図 2.1 簡単なプログラムと等価ハードウェア



(a) 簡単なソフトウェア

(b) 等価ハードウェア

図 2.2 マシン・サイクルごとの動作

二モニツク	機械語	マシンサイクルごとのバスの動作			
LD A, (8000H)	3A	M1	0000番地からの読出し	OPコードの解読	命令読出し ↓ 命令実行
	00	M2	0001番地からの読出し	アドレスの読出し	
	80	M3	0002番地からの読出し		
	—	M4	8000番地からの読出し	命令実行
LD B, 30H	06	M1	0003番地からの読出し	OPコードの解読	命令読出し
	30	M2	0004番地からの読出し	パラメータの読出し	命令実行*
ADD A, B	80	M1	0005番地からの読出し	OPコードの解読→命令読出し→命令実行*	
LD (8001H), A	32	M1	0006番地からの読出し	OPコードの解読	命令読出し ↓ 命令実行
	01	M2	0007番地からの読出し	アドレスの読出し	
	80	M3	0008番地からの読出し		
	—	M4	8001番地への書込み	命令実行
JP 0000H	C3	M1	0009番地からの読出し	OPコードの解読	命令読出し
	00	M2	000A番地からの読出し	アドレスの読出し	↓ 命令実行*
	00	M3	000B番地からの読出し		

* の命令実行にはバス動作を行わない

2.2に示すのは、Z80 CPUのマシン・サイクルを単位として、図2.1(a)のプログラムの実際の動きを示すものです。このように、CPUは命令読出しと命令実行とを交互に繰り返します。また、命令読出しの際には必ずメモリからの読出しを行い、命令実行の際には必要に応じて読出しまたは書込みを行います。

●命令読出し/命令実行

一般に、現在稼動している大半のコンピュータは、まずメモリ（プログラム・メモリ）から1つの命令を読み出して解読し、次にそれを実行します。前者を fetch cycle、後者を execute cycle と呼びます。これはマイコンから大型機に至るまで、どのような複雑なハードウェア、どのような大きなソフトウェアであっても原理的には同じです。*

* ここでは、いわゆるノイマン型でストアード・プログラム（Stored Program）方式のものを指す。

ただし、見掛け上の例外として、8086/88のように命令キュー（Queue）を持っていて命令を先読み（Pre fetch）するものや、大型機に多く見られる多数のパイプ・ライン（Pipe-line）から成るものがある。これらはいずれも、命令読出しと命令実行とを順に繰り返しているようには見えない。しかし、前者の実行ユニット、後者の個々のパイプ・ラインを各々注目すれば、やはり命令読出しと命令実行とを順次繰り返しているに過ぎない。

図 2.3 コンピュータの構造を示す図

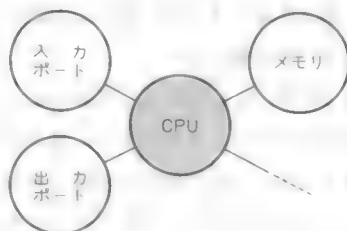
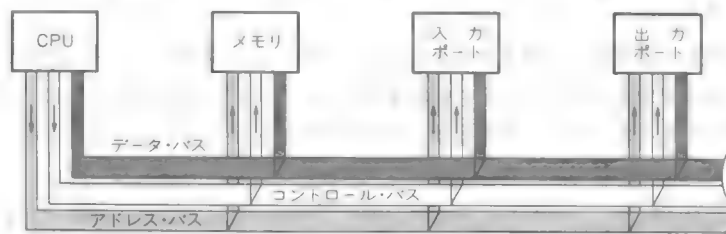


図 2.4 共通バス構成のコンピュータの構造概念図



■バス■説

以上のように、CPU は命令読出しや実行に際して外部のメモリ、入力ポート、出力ポートとの間でデータのやりとりを行います。一般に、CPU が外部に対して行う仕事のほとんどすべてが、このデータのやりとりに関するものだといっても過言ではありません。では、データのやりとりは具体的にどのように行われるのでしょうか。

図 2.3 は、コンピュータの構造を示すのに良く使われる図です。確かに、このように各要素に専用のデータ線を持ったコンピュータもあります。しかし、とくにマイコンやミニコンに類するものでは、共通バス（Common bus：共通母線）構成となっているのが普通です。

図 2.4 は共通バス構成のコンピュータの構造概念図です。このように、どの要素もバスに並列に接続されています。また、アドレス・バスやコントロール・バスは信号の方向が変化しない（常に、CPU から出力される）のに対して、データ・バスは読出し時と書込み時とで信号の方向が異なる点に注目して下さい。

■バスの動作

では、これらのバスを使って、CPU がどのようにデータのやりとりを行うのでしょうか

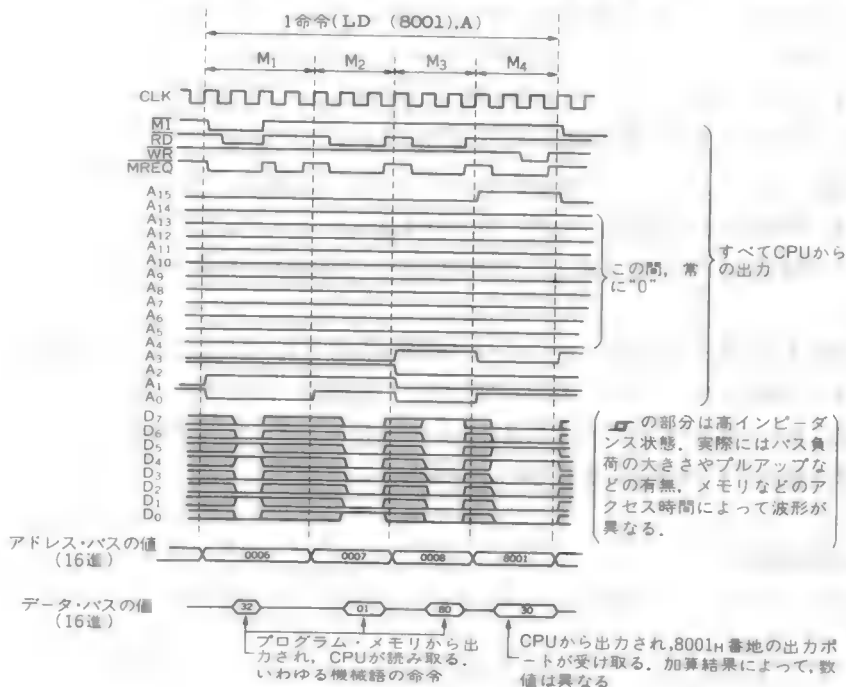
* データのやりとり以外には、たとえば割込みに関するものがある。

か、図2.5に示すのは、図2.1、図2.2のプログラムのうち、LD (8001), A に相当する部分のバス各線の信号です。

まず、この1命令を完了するのに、CPUは4回のバス・サイクルを実行します。バス・サイクルとはデータの読出し、または書込みのことです。各バス・サイクルではアドレス信号を出力し、次に読出しか書込みかを示す信号を出力します。したがって、普通は別々のアドレスに割り当てられているメモリや入出力ポートに対して、同時に読出しや書込みを行うことはできません。

つぎに、バス・スレーブ側を見てみましょう。バス・スレーブには各々アドレスが割り当てられており、自分が指名されたときだけ動作するようになっています。たとえば、図中、M₁~M₃で示す各バス・サイクルではすべてプログラム・メモリからの読出しを行います。アドレス・バス上の信号と、自分に割り当てられたアドレスの値とが一致し、しかも読出しを示す信号がアクティブとなった時にかぎり、プログラム・メモリはデータを

図2.5 LD (8001), A に相当するバス各線の信号



データ・バスに出力します。これ以外の場合は、データ・バスの衝突の可能性があるので、データ・バスにデータを出力してはなりません。

一方、バス・サイクル M_4 では、CPU から出力ポートへの書き込みを行います。書き込まれる側のバス・スレーブ（この場合は 8001H 番地の出力ポート）は、自分が指名されるとともに書き込み信号が印加された場合にかぎり、データ・バス上の値を固定します。出力ポートは単なるレジスタで、最も簡単には D フリップフロップを考えれば良いでしょう。

2.2 マイコン・システムにおけるバス

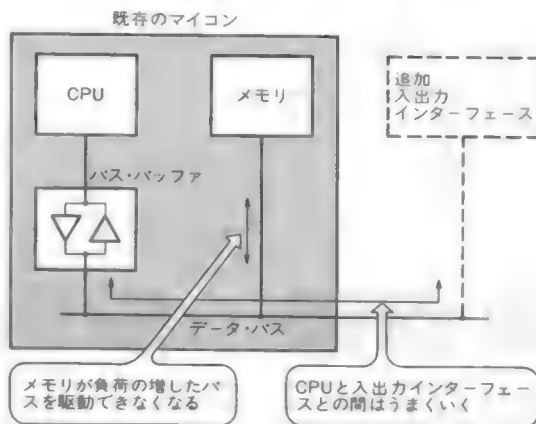
以上でおわかりのように、CPU、メモリ、入出力ポートなど、マイコンの主要な要素はすべてバスに接続されます。マイコン・インターフェースの設計には、バスを正しく利用することが重要です。では、バスを利用するのに必要な事項を検討してみます。

●バス負荷の考え方

バスにはマイコンのほとんどの要素がつながります。とくに、データ・バスには入力、出力の両方の要素が接続されています。

しかし、バスに対して信号を出力できるのは、一度に 1 つの要素だけにかぎられます。そしてこのとき、バスに出力した信号が、他の要素に正しく受け取られることが必要です。

図 2.6 動きそうで動かない例



そのためには、バスに信号を出力する要素が、バスにつながった他の要素を確実に駆動できなければなりません。

よくある設計ミスを図2.6に示します。追加したインターフェースと実際にデータをやりとりするのはCPUだけです。しかし、それ以外の要素にとっても追加インターフェースは負荷となることを忘れてはなりません。

●バス・バッファ

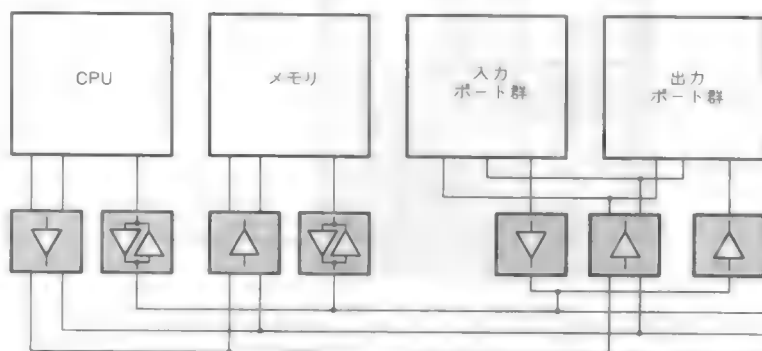
比較的小さなシステムでは、CPUやメモリは直接、バスに接続できます。しかし、バス負荷が増加すると、それを直接には駆動できなくなるのでバッファが必要となります。

図2.7に示すのはバッファの使い方の例です。この例ではCPUやメモリのデータ・バスは双方向性なので、双方向バッファを使っています。一方、入力ポート、出力ポートは別々に一方向のバッファとしています。このように、部分的に一方向のバスを使うほうが回路が簡単になる場合があります。

ちなみに、Z80A CPU (NMOSのもの) では、直流的にはLS-TTL (普通のもの) ならば4個まで駆動できます。一方、バス・タイミングは容量負荷50pFで規定されていますので、MOSメモリのように直流負荷が事実上無視できるものであっても、5個くらいまでにしたほうが良いでしょう。つまり、それ以上の要素を接続する場合はバッファを必要とします。

マイコンで何かをしようとしたとき、ソフトウェア、ハードウェアのいずれか一方だけ

図2.7 マイコン・システム内のバス・バッファの例



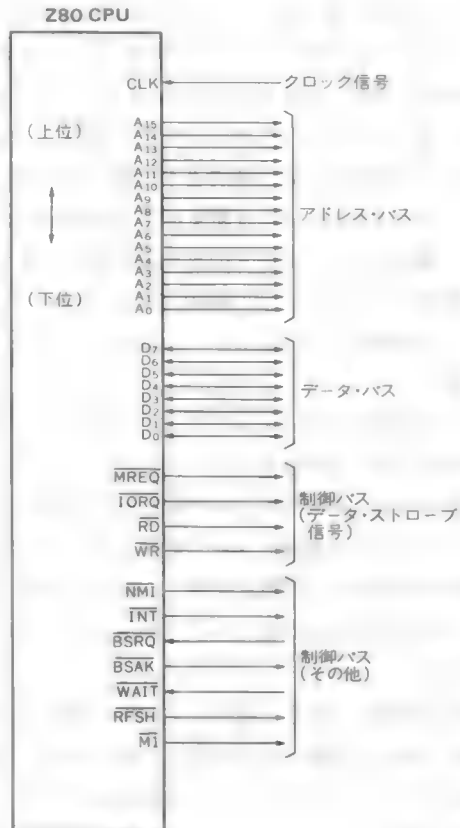
では満足なシステムが作れないことがわかりいただけたと思います。

ハード/ソフトの両面を考えたシステム設計を行ってこそ、マイコンのもつ長所が有効に活用できるのです。

2.3 Z80 CPU バス⁽³⁾

世の中ははやくも 32 ビットの時代に突入しようとしているのですが、Z80 は値段の安さとソフトウェア・ツールの豊富さから未だにコントローラ用として使う機会が少なくありません。

図 2.8 Z80 CPU のバス構成



もちろん、性能上の批判が何かと多いことも、そのバス構成が決して一般的ではなく本項のような基礎的解説に向いていないことも承知のうえで、あえて現場の見地からここではZ80をモデルに話をすすめることにします。

なお、他のCPUについては、次節でその特徴を要約してあります。もちろん、これ以外にもCPUの種類は数多く、筆者としては個人的には捨て難いものもあるのですが、今回は広く知られていないものは割愛させていただきました。

■ Z80 のバス構成

Z80は8ビットのデータ・バスと16ビットのアドレス・バスをもつCPUです(図2.8)。アドレス空間にはメモリ領域とI/O領域とがあり、それぞれ16ビット(64Kバイト)の大きさがありますから、合計128Kバイトが使用できます。

この“メモリ領域”とか“I/O領域”というのはZ80の前身8080から受け継いだソフトの名残りであり、使用する立場から見ればメモリ領域にI/Oポートを接続しようがI/O領域にメモリを接続しようが自由です。名称にこだわることはありません。ただし、メモリ領域とI/O領域とはつぎのような違いがある点に注意してください。

- (1) プログラムはメモリ領域からしか読み出せない。
 - (2) 使用できるアドレス指定方法が異なる。また、両者ともに使用できるBCレジスタ間接アドレスでも、I/O領域に対してはBレジスタ(8ビット)のインクリメント、メモリ領域に対してはBCレジスタ(16ビット)のインクリメントを行うなどの違いがある。
 - (3) 読み書きのタイミングやバス・サイクルの長さが異なる(I/O領域のほうが遅い)。
- 上記のうち(1)はやむを得ず、(2)はソフトウェア上の問題で、ハード設計上知っておかなければならないのはおもに(3)であるといえます。

さて、このI/O領域とメモリ領域とをハードウェア的に区別するには、データ・ストロープ信号($\overline{\text{MREQ}}$, $\overline{\text{IORQ}}$)を使うしかありません。これらの信号がアドレスからかなり遅れて出力される点がZ80のハードウェアの1つの欠点であることはよく知られているところです。

本書ではあまり詳しく触れませんが、Z80には2本の割込み要求端子($\overline{\text{NMI}}$, $\overline{\text{INT}}$)と1本のDMA要求端子($\overline{\text{BUSRQ}}$)とがあります。割込みに関しては割込みアクノリッジ信号が端子として出していない点に注意が必要です。一方、DMAはデータ・バス、アドレス・バスのほか、データ・ストロープ信号($\overline{\text{MREQ}}$, $\overline{\text{IORQ}}$, $\overline{\text{RD}}$, $\overline{\text{WR}}$)がすべて高インピー

ダンスとなります。

■バス・サイクルの■

Z80 は実行する命令により、必要とするマシン・サイクルの数が1～5に変化し、またバス・サイクルも5種類あります。

Z80 の命令実行の例を表 2.1 に示します。

まず、どのような命令であっても、その第1バイト目(OPコード)を読み出すには専用のバス・サイクル(M1サイクルと呼ぶ)を用います。これは、命令の種類を少しでも早くデコードして実行に移すためと、ダイナミックRAMのリフレッシュをこのサイクルで行うためのものです。

命令の第2バイト目以降の読出しとメモリ領域からのデータ読出しとには、メモリ読出しサイクルを用います。一方、I/O領域からのデータ読出しにはI/O読出しサイクルを用います。I/O領域のバス・サイクルの長さをわざわざ変えてあるのは、8080A時代の遅い周辺ICの使用を考えたためです。同様に書込みの際にも、メモリ書込みサイクルとI/O書込みサイクルとの区別があります。

表 2.1 Z80 の命令実行の様子

命令の例	LD A, B	LD A, n	LD (nn), BC	IN A, (BC)
クロック数とマシン・サイクル	1			
	2	M1 OPコードの読出し	M1 OPコードの読出し	M1 OPコードの読出し
	3			
	4			
	5			
	6	MR nの読出し	MR 命令2バイト目の読出し	MR 命令2バイト目の読出し
	7			
	8			
	9		MR	
	10			IOR 読出し
	11			
	12		MR	
	13			
	14			
	15		MW	
	16			
	17		MW	
	18			
	19			
	20			

図 2.9 Z80A CPU のメモリ 読出しサイクル

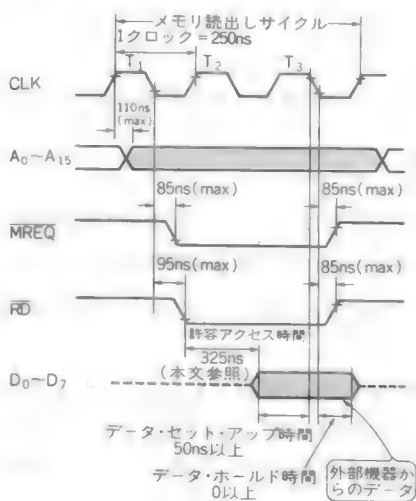
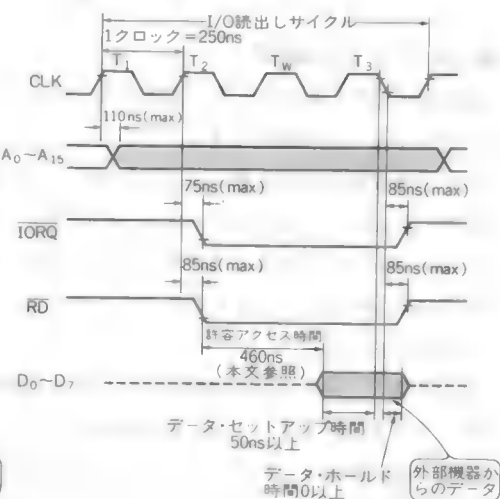


図 2.10 Z80A CPU の I/O 読出しサイクル



では、つぎにこの5種類のバス・サイクルのタイミングを以下に説明します。

(1) メモリ読出しサイクル, I/O 読出しサイクル 図 2.9 にメモリ読出しサイクルのタイミングを示します。なお、ここで説明するのは Z80A をクロック周波数 4 MHz で使用した場合の数値です。

図のいちばん上には書いてあるのは CPU に印加するクロック信号です。普通、入出力インターフェースの設計にはクロック信号を直接使用することはない*ですが、CPU の内部はこのクロックに同期して動いており、タイミング計算上もクロックが基準になります。

メモリ読出しサイクルはクロック信号の立上りから始まります。これに同期して、アドレスも出力されます(ただし、CPU 内部の遅れ時間だけ遅れて確定する)。つぎに、クロックの半周期後に $\overline{\text{MREQ}}$ と $\overline{\text{RD}}$ とが“L”になります。

一般に、入力用インターフェースはこの時点で必要とするすべての情報(アドレスが所定の値か? 指定されたのはメモリ領域か I/O 領域か? 読出しか書込みか?)がそろったことになり、必要とあらばデータの出力を開始します。

CPU が内部にデータを取り込むのは 3 個目のクロック (T_3) の立下りです。したがっ

* Z80 ファミリの周辺 IC は CPU と同じクロック信号を必要とする。

て、すべての情報がそろってからデータを確立させるまでに2クロック分の時間的余裕があることになります。ただし、 $\overline{\text{MERQ}}$ や $\overline{\text{IORQ}}$ が出力されるのに要するCPU内部の遅れ時間や、CPUのデータ取込みに必要なデータ・セットアップ時間を差し引く必要があります。

本書ではこの時間（データ・ストロブ・アクセス時間）をたんにアクセス時間と呼ぶことにします。^{*} 図2.9の場合、入力インターフェースに許されるアクセス時間の最大値は、クロックの立上り・立下り時間を各30nsと仮定すれば325nsとなります。入力インターフェース回路はこの時間内にアドレスおよびデータ・ストロブ信号をデコードし、スリーステート（3ステート）・バッファの出力をアクティブにしなければなりません。

一方、図2.10に示すのはI/O読出しサイクルです。メモリ読出しサイクルと異なるのは、バス・サイクル自体が1クロック分だけ長いこと、 $\overline{\text{IORQ}}$ および $\overline{\text{RD}}$ が2個目のクロック（ T_2 ）の立上りで変化することです。この結果、アクセス時間の許容値は約半クロック分だけ延びて460nsとなります。

なお、Z80 CPUはデータの読み書きが間に合わない場合、WAIT端子を用いてバス・サイクルを延長することができます。しかしTTLを組み合わせてつくった入出力インターフェースでCPUを待たせなければならないことはまれです。むしろ、CPUを待たせないように設計することに留意すべきだと思われます。

(2) メモリ書込みサイクルとI/O書込みサイクル 図2.11に示すのがメモリ書込みサイクルのタイム・チャートです。

読出しの場合と同様に、アドレスはバス・サイクルの最初から出力されます。最初のクロック（ T_1 ）の立下りから、CPUがデータ・バスにデータを出力し始めるとともに、 $\overline{\text{MREQ}}$ 信号が“L”になります。 $\overline{\text{WR}}$ はデータよりも1クロック遅れて“L”になり、さらにその1クロック後に“H”に戻ります。データは $\overline{\text{WR}}$ が“H”に戻ってから半クロックの間、保持されています。

このように、Z80のメモリ書込みサイクルでは $\overline{\text{WR}}$ の前縁に対してもデータ・セットアップ時間が確定されており、出力データ・ラッチとしてDフリップフロップを用いた場合は立上り・立下りのいずれでトリガしてもよいことになります。また、トランスバレント・ラッチを用いることができます。

つぎにI/O書込みサイクルを調べます(図2.12)。メモリ書込みサイクルに比べてバス・

* メモリなどでは、このほかにアドレス・アクセス間隔が重要となる。

図 2.11 Z80A のメモリ書き込みサイクル

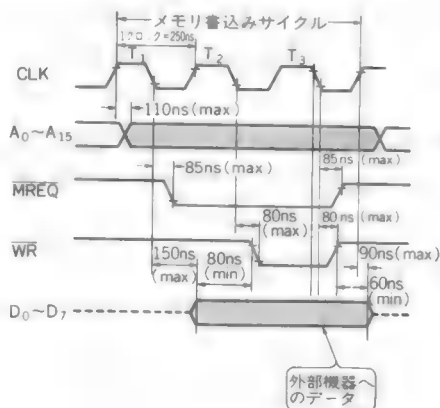
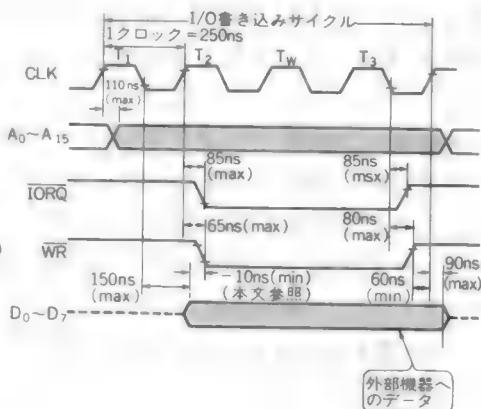


図 2.12 Z80A のI/O 書き込みサイクル



サイクル全体の長さが1クロック分だけ長くなっていることのほか、データが出力されてから \overline{WR} が“L”になるまでの時間が短くなっている点に注意を要します。

このため、 \overline{WR} の前縁に対するデータ・セットアップ時間が最悪値で-10ns、すなわちデータが確定するのが \overline{WR} の前縁より遅くなることがあり得るのです。*したがって、Z80 のI/O 領域に出力ポートを設ける場合、原則として \overline{WR} の後縁でDフリップフロップをトリガすることになります。

(3) M1サイクル 各命令のOPコード読出しの際にのみ使われるのがこのバス・サイクルです。

図2.13に示すように、 \overline{MREQ} および \overline{RD} が通常の読出しサイクルよりも早く“L”になります。CPUがデータを取り込むのは第3クロック目の立上りであり、データ・ストローブ・アクセス時間は215nsときびしいものになります。

ただし、通常は入出力インターフェースでこのサイクルを使用することはありません。また、割込み処理の■に最初にM1サイクルに入りますが、このときはCPUが自動的に2クロックの延長を行うので、アクセス時間の■は少なくなります。

以上、Z80A CPUのバスのタイミングの概略を説明しました。

ハードウェアの設計には以上のような内容を必ず知っておかなければなりません。計算

* Z80Aでもクロック■を低くするか、クロックのデューティ比を変えることにより、セットアップ時間を確保する方法はある。

す、この点、1マシン・サイクルの長さが変化するZ80としばしば対比されます。

また、アドレス空間は16ビット(64Kバイトのみ)で、Z80などと異なりメモリ領域とI/O領域との区別がありません。

▶ 68系バスのタイミング

つぎに、バスのタイミングについて説明します。

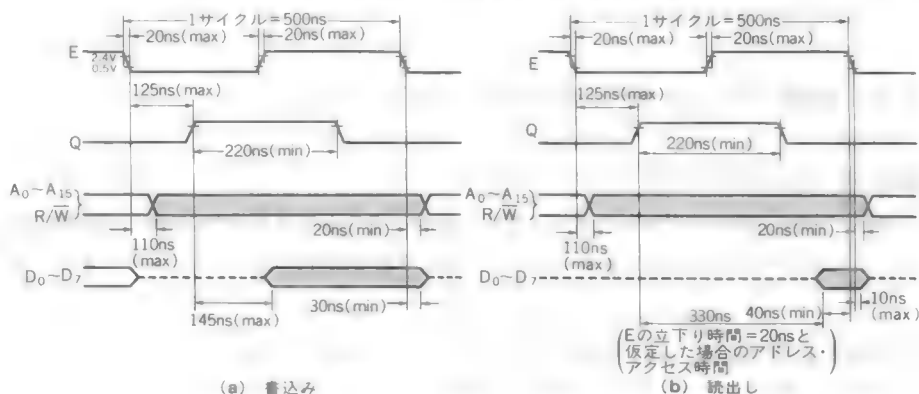
まず、E信号の立下りに同期して16ビットのアドレスが出力されるとともに、同じタイミングで R/\overline{W} (読出し/書込み)信号も出力されます。6802や6800では、アドレス・バス上のアドレスが有効か否か(つまり、そのマシン・サイクルがバスの読み書きを必要とするか否か)を示すVMA(バリッド・メモリ・アドレス)信号が出力されますが、6809ではこの信号はありません。

一方、6809ではE信号よりも位相の進んだQ信号が設けられており、QとEとのORをとった信号をアドレス・ストロブ信号として使用できます。

書込み(CPU→外部機器、 $R/\overline{W} = "L"$)時にはCPUからデータが出力されます。一般にはE信号と R/\overline{W} 信号とにより書込みストロブ信号をつくります。ただし、E信号の立下りから出力データが保持されている時間は30ns(min)と短いので、周辺回路の設計のいかんではデータ・ホールド時間が不足します。6809ではQ信号を用いればなんとかありますが、6800や6802では設計時に慎重な計算を必要とします。

また、読出し(外部機器→CPU、 $R/\overline{W} = "H"$)時には、CPUはE信号の立下りでデータを取り込みます。このタイミングの前後にはデータ・セットアップ時間とデータ・ホー

図2.14 68B09のバス・タイミング(5V±5%, 0~70°C)



ルド時間の両方が必要です。実用上、デコーダやバッファの遅延時間によってデータ・ホールド時間が確保されますが、最悪値設計をするとCPUに対するデータ・ホールド時間が不足することがあるので注意します。

図2.14に68B09をクロック2MHzで動作させたときのタイミング(最悪値)を、図2.15(次ページ参照)に基本的な入出力インターフェースの接続方法をそれぞれ示します。

なお、バス負荷条件は $I_{OL} = 2\text{ mA}$ ($V_{OL} = 0.5\text{ V}$) および $C_L = 130\text{ pF}$ (ただしアドレス $A_0 \sim A_{15}$ は 90 pF) です。

● 8048系(8048~8051/8031/35/39など)のバス⁽⁴⁾

8048シリーズ(RAMおよびROM内蔵の8048~8051, そのEPROM版の $87 \times \times$, ROM外付けの $803 \times$)はインテル社の代表的なシングルチップ・コンピュータで、CMOSタイプも含めて多くのセカンド・ソースがあり、モステック3870系と並んで非常に広く使用されています。

このCPUは、外部メモリや周辺装置が不要の場合は24本の端子を入出力ポートとして使えますが、外部メモリなどを接続する場合はその一部(たとえば12本)をバスとして使用します。その場合、8本はデータとアドレスとの時分割バスとなっており、結局データ8ビット、アドレス12ビットとして使用することになります。また、プログラム・メモリ領域とデータ・メモリ(およびI/O)領域とが分かれているので注意が必要です。

■ 8048系バスのタイミング

まず、命令読出し時(図2.16(a))には、バスにアドレスが出力されますので、ALEの立下りでこのアドレスをラッチして使用します。ラッチはトランスバレント・ラッチでもDフリップフロップでもかまいませんが、トランスバレント・ラッチのほうが t_{AL} (11MHz発振時に50ns)だけアクセス時間を長くとれます。つぎに、命令読出しストロープ信号として $\overline{\text{PSEN}}$ が“L”になりますから、プログラム・メモリからデータ(プログラム)を出力します。CPUは $\overline{\text{PSEN}}$ の立上りでこれを取り込みます。

一方、データ・メモリや外部I/Oポートの読出し時を図2.16(b)に示します。この場合は、読出しストロープ信号として $\overline{\text{RD}}$ が“L”になります。先程の $\overline{\text{PSEN}}$ とこの $\overline{\text{RD}}$ とは出力されるタイミングが少し異なっており、 $\overline{\text{RD}}$ に対するアクセス時間のほうが余裕があります。

最後にデータ・メモリや外部I/Oポートに対する書込み時を図2.16(c)に示します。書込みストロープ信号 $\overline{\text{WR}}$ は、 $\overline{\text{RD}}$ と同じタイミングで出力されます。 $\overline{\text{WR}}$ の立下り時には

CPUからのデータはまだ出力されていませんので、 \overline{WR} の立上りを用いて書込みを行わなければなりません。

なお、図2.16に示したのはインテル社(オリジナル・メーカー)のHMOS版の8048(8048 HL-1)を11MHzで使用したときのタイミングです。相当品種もクロック発振周波数が同じならば大体似たようなタイミングですが、NMOSかCMOSかにより、あるいはメーカーにより、細かい数値は相当異なりますので、実際に使用するときは注意が必要です。

バスの負荷は $PSEN$, \overline{RD} , \overline{WR} , ALE が $I_{OL}=1.8mA$ ($V_{OL}=0.45V$) および $C_L=80pF$, その他のバスは $I_{OL}=2mA$ ($V_{OL}=0.45V$) および $C_L=150pF$ です。基本的な入出力方法の例(普通、この種のシングルチップ・コンピュータでこのような回路を外付けする必要はないが…)を図2.17に示します。

図2.15 68B09と入出力ポートとの接続

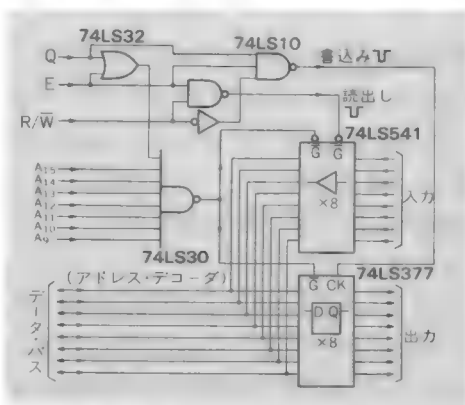
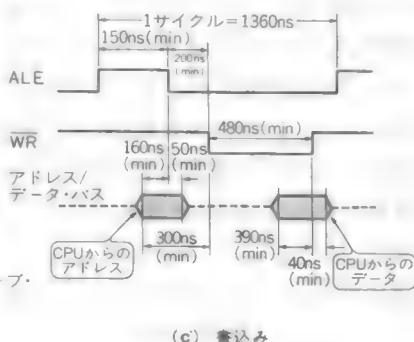
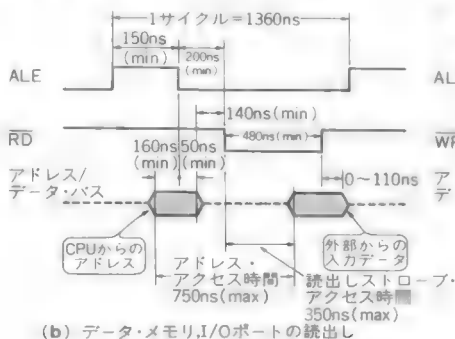
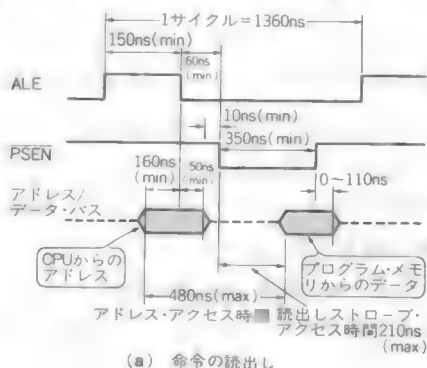


図2.16 8048のバス・タイミング
(5 V \pm 10%, 0~70°C)



● Z8系 (86××) のバス⁽³⁾

ザイログ社のシングルチップ・コンピュータ Z8 シリーズは、Z80 などの汎用 8 ビット CPU に比べて機能的に一段上であり、8 ビット系のコントローラとして今後が期待されています。

命令は 16 ビットの Z8000 のそれに似ており、バス構成も Z8000 とともに“Z-BUS”と呼ばれています。ただし、Z8 はもともとシングルチップ・コンピュータですので I/O 端子の一部をバスとして使用する点は 8048 などと同じです。

アドレス空間はデータ領域と命令領域とに分かれており、各々最大 64K バイト (合計

図 2.17 8048 と入出力ポートとの接続

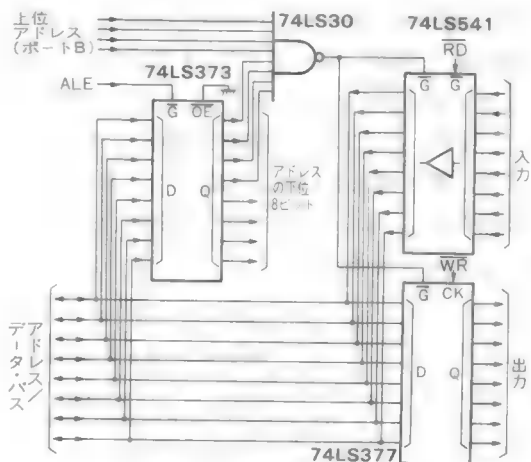
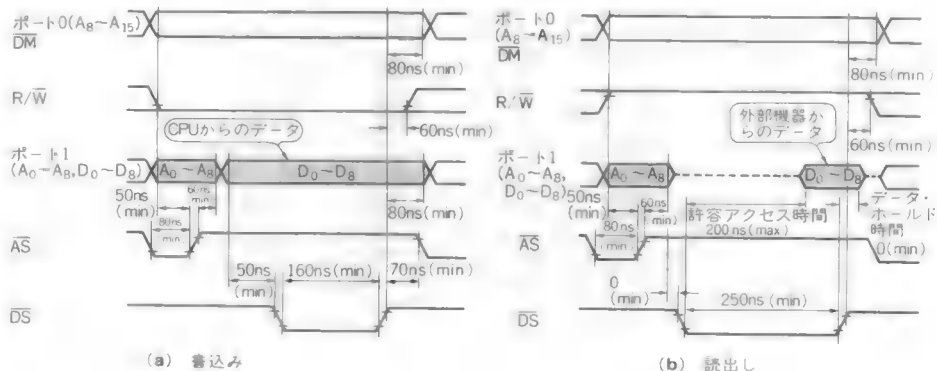


図 2.18 Z8 のバス・タイミング



128K バイト) が使えます。

■ Z8系バスのタイミング

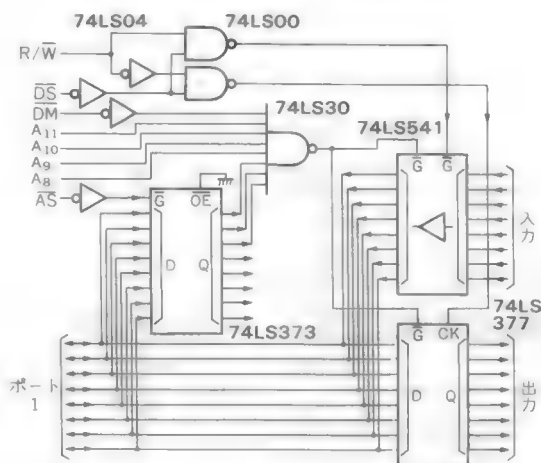
図 2.18 (a) に書込みタイミング, 同図(b)に読出しタイミングをそれぞれ示します。バス・サイクルが始まると、ポート 0 に上位 4 または 8 ビット、ポート 1 に下位 8 ビットのアドレスが出力されます。同時に \overline{AS} (アドレス・ストローブ) が“L”になりますので、トランスパレント・ラッチを用いて下位 8 ビットのアドレスをラッチします (D フリップフロップも使用可能)。 R/\overline{W} (読出し/書込み) および \overline{DM} (データ領域を示す) はアドレスと同時に出力されます。このあと、 \overline{DS} (データ・ストローブ) が“L”になるので、データの転送を行います。

\overline{DS} は書込みの場合のほうが読出し時に比べて遅く“L”になり、書込みデータのラッチには前縁・後縁のいずれでも使えます。また、トランスパレント・ラッチも使用できます。

一方、読出し時は \overline{DS} が早く“L”になり、アクセス時間が少しでも長くできるようになっています。許容アクセス時間は 200ns です。

なお、図 2.18 は Z8 MCU (Z86××) をクロック 8 MHz で使用した場合のもので、バス負荷条件は $I_{OL} = 2\text{ mA}$ ($V_{OL} = 0.4\text{ V}$) および $C_L = 150\text{ pF}$ です。基本的な入出力の例を図 2.19 に示します。

図 2.19 Z8 と入出力ポートとの接続



8088 (ミニマム・モード), 8085, NSC 800 のバス (4)(16)

8085 はインテル社の8ビットCPUで8080Aの後継機、8088は8086と命令互換性のある8ビットCPU、NSC800はZ80と命令互換性のあるNS社のCMOS 8ビットCPUです。

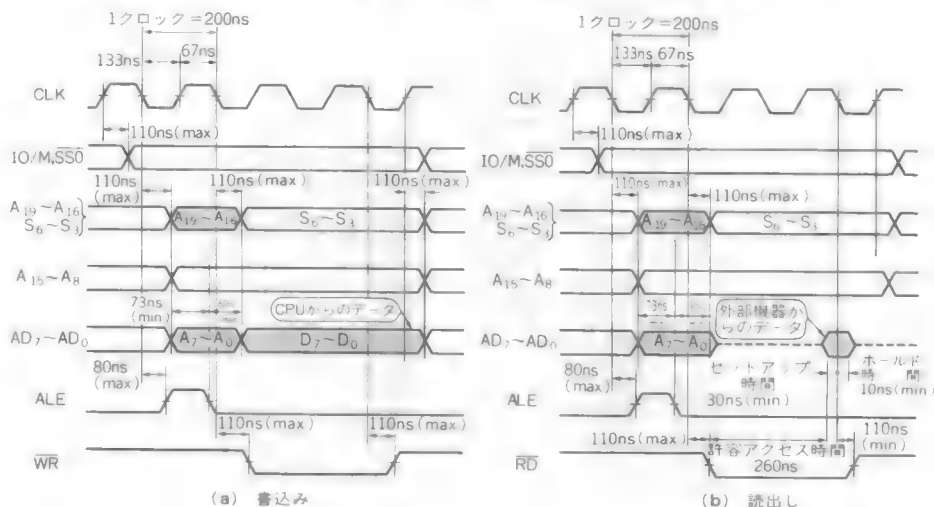
8088にはミニマム・モードとマキシマム・モードとがありますが、そのうちミニマム・モードを用いれば、この3種のCPUは非常によく似たバス構成となっています。ここでは8088を例にとって、そのタイミングを説明します。

8088 (ミニマム・モード) は20ビットのアドレスと8ビットのデータをもち、下位8本のアドレスはデータと、上位4本のアドレスはステータス信号とそれぞれ多重化されています。

8088, 8085, NSC 800 のバスのタイミング

図2.20(a)に示すのは書込みのタイミングです。8088のクロック信号は最適デューティ比が1/3で、その許容範囲が狭いのが難点です。各時分割バスのアドレス($A_{19} \sim A_{16}$, $A_7 \sim A_0$)は、ALE (アドレス・ラッチ・イネーブル) 信号でラッチします。 $A_{15} \sim A_8$ は時分割ではありませんが、インテル社ではラッチを付けることを推奨しています。書込みストローブ \overline{WR} の前縁(立下り)ではデータは保証されませんから、必ず \overline{WR} の後縁(立上り)

図2.20 8088(ミニマム・モード)のバス・タイミング(5 V \pm 10 %, 0 \sim 70°C)



でDフリップフロップにデータを書き込みます。

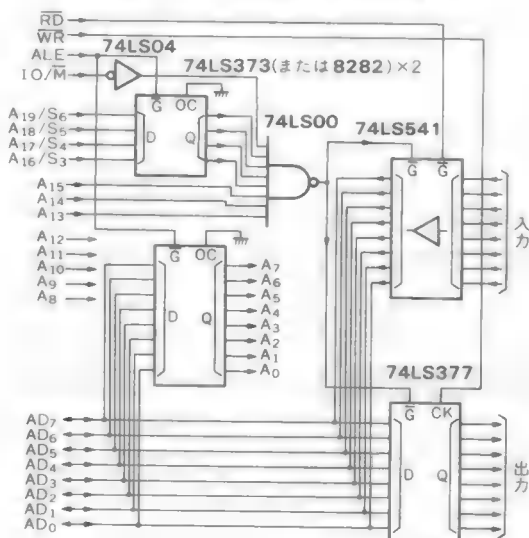
一方、図2.20(b)に示すのは読出し時のタイミングです。 \overline{RD} 信号は \overline{WR} 信号と同じタイミングで出力されます。 \overline{RD} からの許容アクセス時間は約260ns(クロック5MHz)です。

なお、図2.20に示す数値はクロック周波数5MHz、デューティ比1/3、立上り、立下り時間は無視したものです。バス負荷は $I_{OL}=2\text{mA}$ ($V_{OL}=0.45\text{V}$)および $C_L=100\text{pF}$ です。基本的な入出力ポートの接続を図2.21に示します。

以上、いくつかの8ビットCPUのバス・タイミングを説明しました。これ以外にも多くのCPUがありますが、ハードウェアの詳しい資料が入手できれば、入出力インターフェースの設計は容易でしょう。

一方、今後は16ビットや32ビットのCPUが制御用にも使われるようになるでしょうが、基本的には何ビット機であってもハードウェアにかぎってみれば同じことです。ただし、16ビット以上のCPUでは、1語(たとえば16ビット)のうちいずれかの8ビットだけを扱う場合と、1語全部を扱う場合とがあり、そのための制御信号の出方に各CPUの特徴が見られます。

図2.21 8088と入出力ポートとの接続



また、CPU が高速になるにしたがい、メモリや入出力インターフェースがアクノリッジ信号を返さないかぎり、CPU が待ち続ける回路構成を採用することも増しており (Normally-Wait などとも呼ぶ、68000 では基本的にこの形)、インターフェースの設計にも多少の考慮が必要です。

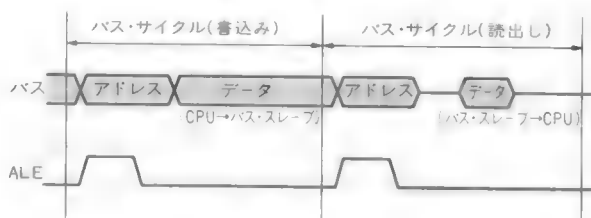
しかし、いずれの場合も CPU の動作をよく理解すれば、設計はきわめて容易だという点は覚えておいてください。

コラムB バスの分離

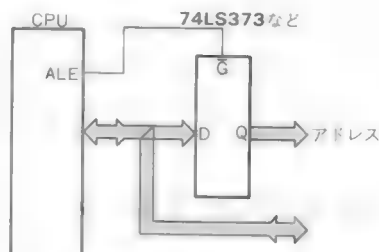
●データ/アドレス/コントロール

普通、CPU のバスはデータ・バス、アドレス・バス、コントロール・バスなどと分類します。いうまでもなく、読み書きするデータを乗せるのがデータ・バス、アドレスを指定するのに使うのがアドレス・バス、読出しや書込みのストロブ信号や割込み線などをコントロール・バスと呼びます。

図 B.1 時分割バスとその分離

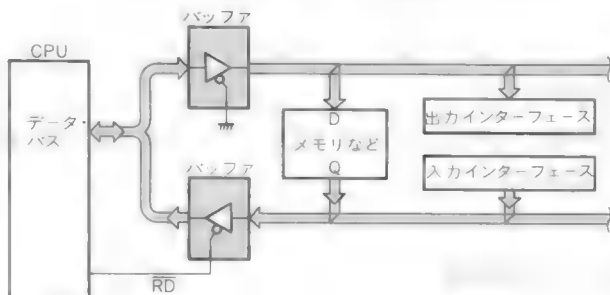


(a) タイミング



(b) ALE信号の使用方法

図 B.2 一方向のバスの1例



■時分割（多重化）バス Multiplexed bus

CPU はすでにご承知のように、バスの本数が非常に多く、さらに高性能の CPU ほどその本数が増える傾向にあります。そこで、配線の本数を減らすため、バスの時分割（多重化）が行われます。

最も多いのはデータとアドレス（の一部）との多重化です。これは図 B.1 のように、バス・サイクルの初めにはアドレスを乗せ、そのあとはデータ・バスとして使うものです。図 B.1 のように、ALE 信号を用いてラッチすれば、アドレス信号を分離することができます。

■双方向データ・バス

現在のマイコン・システムでは、データ・バスは読み書きに共通に使う（双方向）のが普通です。しかし、以前は読出し用と書込み用との2組のデータ・バスを持つシステムはめずらしくありませんでした。

今でも、入力と出力とが別々の端子になっているメモリを使う場合など、部分的に専用のバスを使ったほうが、ハードウェアが簡単になることがあります(図 B.2)。また、多数の入出力を持つシステムでは、入力ポート専用、出力ポート専用といったバス・バッファ方法は一考に値します(詳しくは第4章、例 4-3などを参照して下さい)。

コラムC バス・マスタとバス・スレーブ

バスには様々な要素が接続されますが、そのうちバスを管理するものをバス・マスタと呼びます。通常、アドレス信号や制御信号はバス・マスタが出力します。一方、

バス・マスタ以外のものをバス・スレーブと呼びます。

簡単なマイコン・システムでは、CPU がバス・マスタであり、メモリや入出力インターフェースなどがすべてバス・スレーブです。

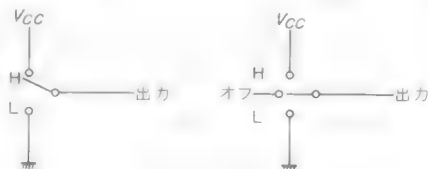
しかし、DMA コントローラやコプロセッサを含むシステム、あるいは複数 CPU システムでは、バス・マスタとバス・スレーブとの関係は複雑になります。

コラムD スリーステート

計算機のデータ・バスのように同じ信号線に何本もの出力線が接続される場合、2 本以上が同時にバスに信号を出力すると衝突が起こるので、不要な出力をオフ状態にしなければなりません。通常の論理値 (H, L) のほかに第 3 の状態 (オフ) を持つので、この機能をスリーステート (three-state) と呼びます (トライステートは商標)。

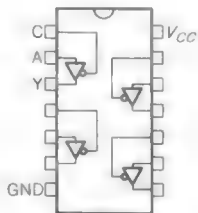
スリーステート機能は、図 D.1 のようなものだと考えればわかりやすいでしょう。TTL では 74LS125, 126 が有名です (図 D.2 (a))。また、4 ビットまたは 8 ビット単

図 D.1 スリーステート機能の概念

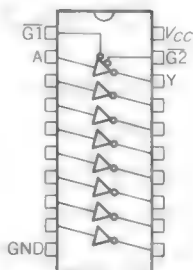


(a) 通常のデジタル出力 (b) スリーステートの出力

図 D.2 スリーステート出力の IC の 1 例⁽⁵⁾



(図は 74LS125. C = "H" で出力
オフとなる。
74LS126 は C = "L" で出力オフ)
(a) 74LS125/126



(図は 74LS540. G1 = H または G2 = H
で出力オフ。
74LS541 は非反転)
(b) 74LS540/541

表 D.1 駆動能力の比較⁽⁵⁾

		通常のゲート (74LS00)	スリーステートのゲート (74LS125A)	専用のバス・ドライバ (74LS540)	
$I_{OH(max)}$	$V_{OH} \geq 2.7V$	-0.4			mA
	$V_{OH} \geq 2.4V$		-2.6	-3	
	$V_{OH} \geq 2.0V$			-15	
$I_{OL(max)}$	$V_{OL} \leq 0.4V$	4	12	12	mA
	$V_{OL} \leq 0.5V$	8	24	24	

位のスリーステート・バッファもあります。その1例として74LS540、541の端子接続を図D.2(b)に示します。

一般に、スリーステート出力はバスに接続されることが多いので、駆動能力も大きく目になっています。表D.1にその1例を示します。なお、TTLの場合、スリーステート機能を持つものは消費電力がかなり大きく、しかも出力がオフ状態のときに消費電力が最大となります。消費電力の計算にはくれぐれもご注意ください。

一方、複数の出力を並列に接続する方法として、オープン・コレクタ（またはオープン・ドレイン）もあります。以前はよく使われたものですが、速度が遅いなどの欠点があり、バスに接続するものとしては、スリーステートに置き換えられてしまいました。ただし割込み信号については、オープン・コレクタ（またはオープン・ドレイン）のワイヤードOR機能を生かして利用されることが少なくありません。

コラムE バス・タイミングの記述方法

実際のバスの各信号は図2.5(24ページ参照)のように変化します。しかし、ICのデータ・シートには図E.1のような書き方がされています。

図中、④の部分はアドレス信号(16本)の最大遅延時間です。ただし、このように書いてあっても、16本のアドレス信号がすべて同時に変化するものではありません。実は、図E.2のように立上りまたは立下りのタイミングが0~△△nsの間にバラついているのです。

一方、データ・バスは出力と入力（スリーステート時を含む）との両方の状態があ

図 E.1 データ・シートに記載されているタイム・チャートの例

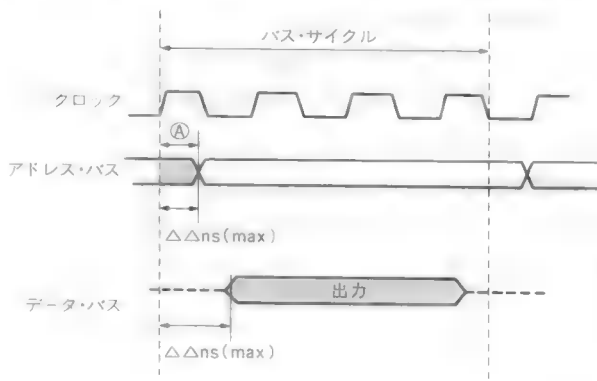


図 E.2 図 E.1 (A) 部の実際

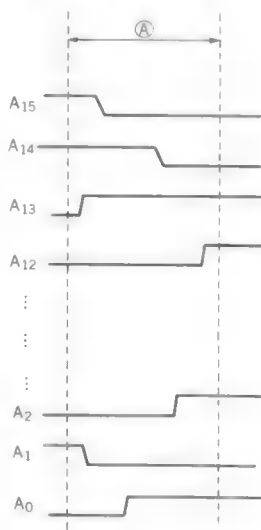
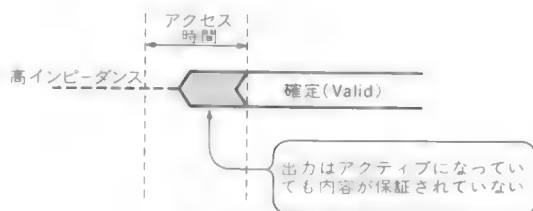
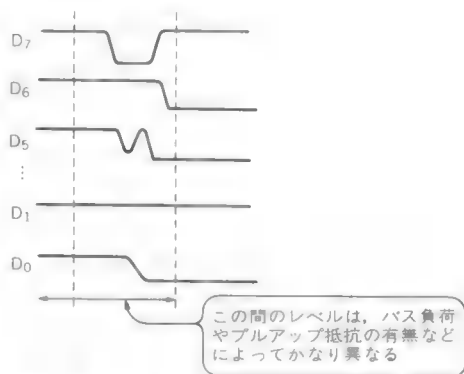


図 E.3 スリーステート出力タイミング



(a) データ・シート上の記載例



(b) 実際の信号

ります。入力の場合、データ・セットアップ時間およびデータ・ホールド時間が必要です。これらの時間は“最小限これだけが必要”という値で示されます。一方、出力の場合、データ・バスの各信号がオフ状態からアクティブになり、そのあとデータが確定します。このタイミングも最悪のケースでは図 E.3 のようにバラついていると考えられます。

デジタル回路では、このように最悪値設計が行われますが、注意したいのはその特性の条件です。とくに電源電圧範囲、温度範囲、負荷条件によりタイミング特性は異なります。TTL の大部分が 25°C でのタイミングしか保証していないのは、本当は困りものです。

第 3 章

マイコン入力インターフェースの実際

ここでは外部機器からマイコンにデータを入力する方法を示します。

まず、この入力用インターフェースの一般形を図 3.1 に示します。外部機器からのデジタル信号は、①レベル・シフトによって本来のレベルおよび形態（たとえば接点信号）から所定のレベル（たとえば TTL レベル）に変換し、②必要に応じてラッチやデコードを通し、③バス・インターフェース部分を経て、④マイコンのバスに接続します。アナログ信号の場合、適切なレベルに変換してからアナログ-デジタル変換器（A-D コンバータ）によりデジタル信号にかえ、あとはデジタル信号として処理されます。

この章では以上の各部について事例を紹介しながら設計法を説明します。

3.1 入力信号の種類と処理法

●入力信号の形態とレベル

マイコンに接続する入力信号は、アナログ信号とデジタル信号とに大別できます。マイコンがアナログ信号を取り込むには、コンパレータなり A-D 変換器なりを通してデジタル信号に直してから、デジタル信号として入力することになります。

デジタル信号として使用する機会が多いのは TTL レベルや接点、あるいはオープン・コレクタなどの信号です。ただし、細かく見れば、たとえ TTL レベルであっても許容ファン・アウトの違いから、接続可能なものとそうでないものとが生じます。また、接点信号はチャタリング特性の違いや、接点の種類（微小信号用か否か）などにより回路上の工夫が必要です。

RS-232C、RS-422 や、IEEE-488 (GPIB) のように規格が定まっているものに対して

図 3.1 入力用インターフェースの一般形

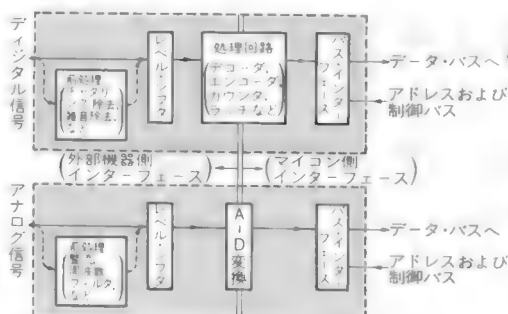
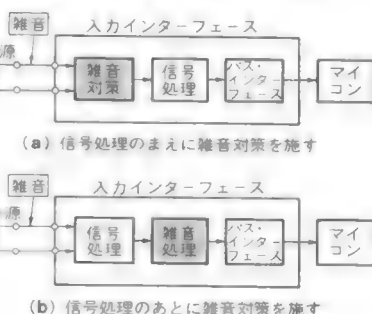


図 3.2 雑音対策の挿入位置



は、専用のレベル変換 IC が使えます。それ以外のものとして CMOS, PMOS, HTL, リレー論理回路など、レベルの違うデジタル信号があります。これらは個々にレベル変換回路を設ける必要があります。ただし、ハードウェアの標準化という観点から、なるべく広い範囲のレベルに対して使用できる回路を考えておくのは有効です。

以上の問題とは別に、先に触れた雑音の問題により、アイソレーション（絶縁）や雑音処理を行わなければならない場合があります。一般に、このような処理はなるべく信号源に近い部分で行ったほうが望ましいといえます（図 3.2 参照）。

一方、アナログ記号は各種のセンサやトランスジューサ、あるいは測定機器からの信号と考えられます。簡単なものではコンパレータで“H”か“L”かに分けてしまえばよいのですが、一般的には A-D 変換することになります。

A-D 変換器自体は非常に高性能のものが入手できますが、マイコンという大きなデジタル回路と共存する点が問題となります。たとえば、12 ビットの A-D 変換を行うと、アナログ信号の分解能は $10\sim 100\mu\text{V}$ という微小なものになり、わずかなデジタル雑音の混入も大問題となります。

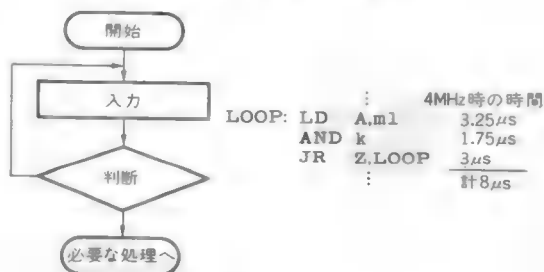
以上のように、入力信号の形態およびレベルの違いが、入力インターフェースの外部機器側の設計に大きく影響します。

●入力信号の速度

マイコンが信号を入力する速度には、少なくともつぎの2種類があります。

- (1) 信号が印加されてからそれを検出するまでの応答時間。
- (2) 信号を処理し終わるまでの処理時間、あるいは連続的に処理を繰り返せる処理周期。

図 3.3 信号の判断(ポーリング・ループ)



マイコンに接続する信号のなかには、何年かに一度しかアクティブにならない（あるいは永久に使用されることのない）異常検出信号のようなものがあります。ただし、異常発生後1秒以内に対策をとらねばならないとすれば、少なくとも応答時間は1秒以下でなければなりません。しかし、処理周期はこれほど短い必要はありません。

一方、FIFO (First-In First-Out: 先入れ先出しメモリ) を介して入力されるデータ列を読み込むような場合、平均的な処理周期がデータ転送速度よりも速いことが重要です。

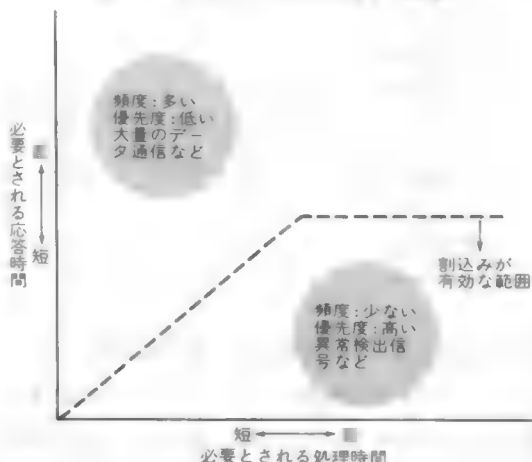
よく知られているように、信号の有無を知るには割込みとポーリング*との2種類があります。先程の区分から見ると、割込みを用いれば応答時間を早くすることができます。ただし、処理時間については大差ありません。ちなみに、ソフトウェアで達成できる応答速度はどの程度でしょうか。簡単なループを図3.3に示しますが、4MHzのZ80Aでこのループを1回まわるのに8 μ Sが必要です。ループ内に処理や判断が追加されれば、これより遅くなります。

一方、割込みの場合はどうでしょう。一般に8~16ビットの汎用CPUで多く使われている割込み方法では、CPUが割込みを受け付けてから実際の処理ルーチンに入るまでにつきのような手順が必要です。

- (1) 実行中の命令のアドレスをスタックに退避する。
- (2) 外部機器または割込みコントローラから割込みベクタ（あるいは機器番号）を入力する。
- (3) 割込み用のエントリ・ポインタ・テーブルを参照して処理ルーチンを呼ぶ。

* ここでいうポーリングとは、ソフトウェア・ループで信号を判断することをいう。本来は複数の信号を順々に調べることをポーリングという。

図 3.4 応答時間と処理時間との関係



このほか、必要なレジスタの退避を含めると、一回の割込みについて数 $10 \sim 100 \mu\text{s}$ くらいのオーバーヘッドが必要です。つまり、応答時間は $100 \mu\text{s}$ のオーダとなり、ポーリング・ループと大差ありません。ただし、CPU が他の仕事をしていてポーリングができない場合でも割込みは可能ですから、やはり応答時間を重視する場合は有効です（図 3.4 を参照）。

このほか、多くの CPU には DMA（ダイレクト・メモリ・アクセス）機能があります。ただし、これは機能とはいっても、CPU が仕事を休んで仕事場を他人に貸しているのですから、外部ハードウェアの問題となります。

以上に説明した速度よりも速い処理が必要な場合、インターフェースの外部機器側ハードウェアの出番となります。また、遅い信号であっても様々な理由からハードウェアとしたほうがよい場合もあります。これらについては本章 3.3 で詳しく説明します。

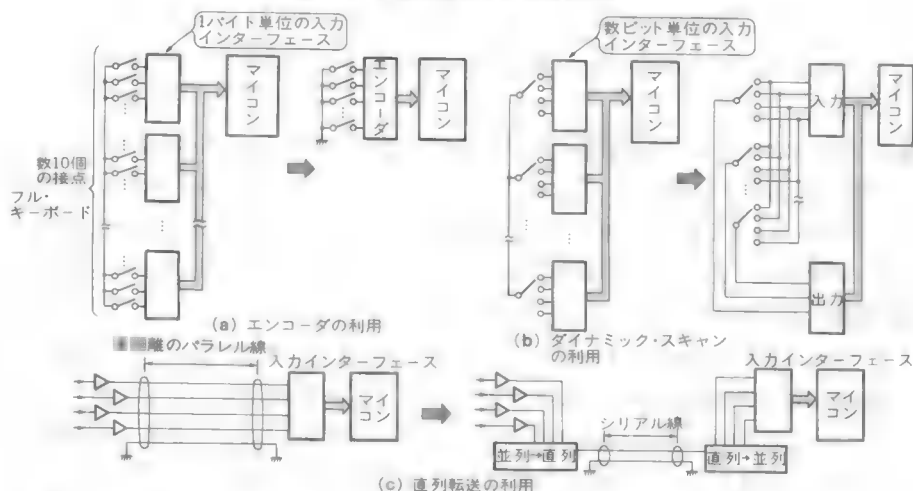
●入力信号の数と処理方法

入力信号の数が多くなった（数 10 本以上）場合、少数の信号（たとえば 8 本 = 1 バイト）の入力用インターフェースをたくさん作って並列に接続しただけではいけないのでしょうか？

もちろん、ちゃんと設計されたものならば何回も接続しても動作するはずですが、その場合につぎの二つの問題が生じます。

- (1) ハードウェア上の問題……多くの回路を接続することにより、バス負荷や消費電力

図 3.5 多数の信号を取り込む工夫



が増加し、設計が困難となる。

(2) ソフトウェア上の問題……前項で述べた入力速度は、入力信号数を増やせばさらに低下する。また、個々の処理も複雑となる。

これらの問題に対して、つぎのような対策がとれないかを考えてみます。

- (1) エンコード（符号化）あるいは簡単な論理演算で信号数が減らせえないか？
- (2) ダイナミック・スキャンにより信号数を減らせえないか？
- (3) 優先順位がつけられないか？
- (4) 直列（ビット・シリアル）に変換できないか？

このうち(1)の方法によりバス・インターフェースを簡単にできます（図 3.5(a)）。また、(2)の方法は外部機器側の配線を少なくします（図 3.5(b)）。たとえば 50 個以上の接点をもつフル・キーボードでは(1)、(2)を併用しているのが普通です。また、多数のロータリ・スイッチの読込みには(2)の方法が、遠距離の信号源を見るには(4)の方法がそれぞれ有効です（図 3.5(c)）。

これらの事例については、このあと 3.2 と 3.3 とで詳しく説明します。

3.2 入力インターフェースのマイコン側の設計

入力インターフェース回路が必ずしも二分できるわけではありませんが、図 3.1 に書い

たように、マイコン側と外部機器側とに分けて考えてみるのが便利です。ここでは、まずマイコン側について調べます。

入力インターフェースのマイコン側の部分が受け持つのはつぎのような点です。

●マイコンのデータ・バスを駆動するスリーステート（3ステート）・バッファ

一般には、何らかの情報（デジタル・データ）をデータ・バスに乗せることになります。データ・バスはCPUやメモリ、あるいは他のインターフェース回路が時分割で信号を出力しますから、必要とされる時点以外はスリーステート（高インピーダンス）となって他の回路のじゃまにならないようにします。

データ・バスへの接続にはオープン・コレクタを使う方法もありますが、ICが安価となった現在、ごく簡単な用途以外ではあまりメリットがありません。

データ・バスは多くの汎用CPUではTTLレベルを満足することが必要です。駆動能力は、バスにおらさがっているすべての負荷から計算できます。また、アクセス時間はデータ・ストロブ信号（ \overline{RD} など）に対して100~500nsのものが多く、TTLのスリーステート・バッファを使えば、たいいていは要求を満足できます。

ただし、特殊なレベルや速度を要求するCPUもありますから、注意が必要です。

●アドレスや制御信号が所定の組合せとなったことを検出するデコーダ

上記のスリーステート・バッファの出力をアクティブにするのは、アドレスや制御信号が所定の組合せになった時点だけです。これを検出するためデコード回路が必要となります。

CPUのアドレス領域を有効に使うにはフル・デコード（コラムG）とすべきですが、実際には必要に応じて単純化します。また、デコーダの設計によって取込み方法（バイト単位かビット単位か、など）が決定されるわけで、ソフトウェアとの兼ね合いも考えます。

デコーダはCPUのアドレス・バスや制御バスの負荷になるとともに、アクセス時間に影響する部分ですから、タイミング設計時には注意が必要です。

例 3-1 基本的な入力インターフェース

マイコンへの入力は、データ・バスに信号を乗せることが基本となります。これにはアドレスや制御信号が所定の値となったことを検出するデコーダと、必要なときだ

けデータ・バスに信号を乗せるスリーステート・バッファとを用います。

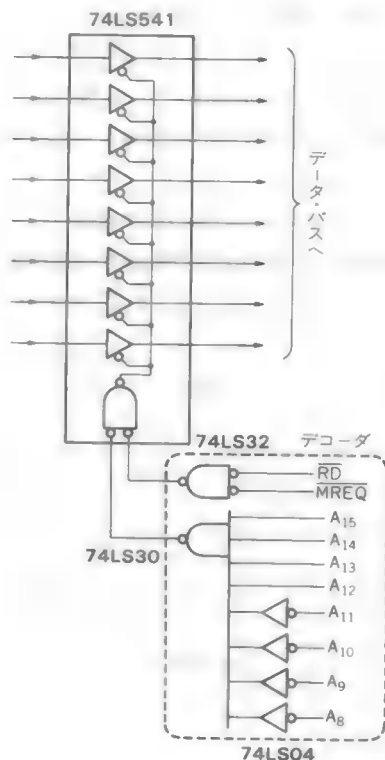
様々な外部機器からの信号（アナログ・デジタルを問わない）は適切に処理したうえで上記の回路に接続すれば、基本的にはデータの読み込みが可能となるわけです。本例ではこの基本的なバス・インターフェースの設計方法を説明します。

▶ **8ビットの信号を1バイトとして並列に取り込む回路** 入力用のバス・インターフェース部分の例を図3.6に示します。

バス・インターフェースとしては、バスに接続できるようなスリーステートの出力をもち、所定のアドレスおよび制御信号が印加されたときのみ出力をアクティブにするデコーダ（コラムF、G参照）を付加する必要があります。

同図は8ビットの入力信号（TTLレベル）を1バイトとして取り込むものです。CPUは

図3.6 入力用バス・インターフェースの例



Z80 を仮定しています。デコーダは特定の信号の組合せを検出すればよいのですから、様々な方法が考えられます (コラム F)。

▶マイコンのバスに接続するためには つぎに、この回路がマイコンのバスに実際に接続できるかどうか調べてみましょう。これは、つぎの2項目に分けて考えてみます。

- (1) この回路を付加することにより、CPU や他のバス・スレーブなどマイコン内部に与える影響。
- (2) この回路がバスを駆動できるか否か。

まず、前者はバス負荷の増加分として表されます。図 3.6 の回路を例にとると、データ・バスについては 74LS541 の出力漏れ電流 (I_{OL} , I_{OH}) と出力容量および配線などの容量とが付加されます。74LS541 の I_{OL} , I_{OH} はそれぞれ $20\mu\text{A}$ (max) です。出力容量は TTL の場合、規格表には明示されていないことが多いのですが、 2pF 程度と考えて良いでしょう。配線の容量は、普通のプリント基板内で長さ 50mm 以下ならば 5pF 以下と考えます。ただし、IC ソケットを使うとこの値は少し増します。その量は 1pF を目安とします。以上の様子を図 3.7 に示しますが、一般的に、スリーステートのバッファ (ここでは 74LS541) を 1 個だけデータ・バスに付加しても、データ・バスにとってとくに重い負荷にはなりません。

一方、アドレス・バスおよび制御バスへの影響を考えてみます。これはデコーダ回路の入力電流と入力容量および配線などの容量との追加になります。図 3.6 の回路では、アドレス・バスのうちの $A_{15} \sim A_8$ と $\overline{\text{RD}}$ および $\overline{\text{MREQ}}$ とに TTL の入力が接続されます。接続

図 3.7 インターフェース回路がバスに与える影響

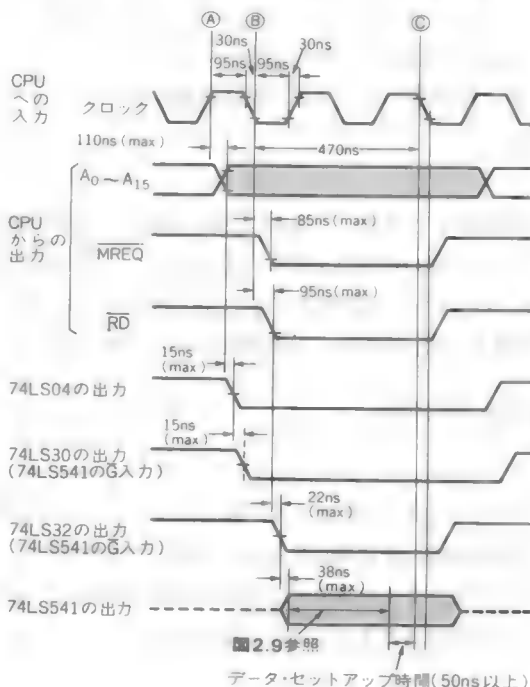


されるのはすべてTTLのゲートであり、 $I_{IL} = -400\mu\text{A}$ (max), $I_{IH} = 20\mu\text{A}$ (max) です。入力容量および配線の容量はそれぞれ2pFおよび5pFを目安とします。Z80 CPUの直流的バス駆動能力は“L”レベルで1.8mA, “H”レベルで $250\mu\text{A}$ ですから、直流的にはLS-TTLを4個まで駆動できます。したがって、バスに接続されている他の回路との合計が規格内ならば、CPUから図3.6の回路の駆動は可能です。

つぎに、先程の(2)の問題を考えてみます。これには静的なバス駆動能力とタイミングの問題とがあります。74LS541の駆動能力は“L”レベル(0.4V)では12mA, “H”レベル(2.4V)では3mAですから、CPUが直接に(バッファなしに)駆動可能なバスならば、十分に駆動可能です。

▶**タイミングの検討** では、タイミングの問題はどうでしょう。図3.6の回路の場合、アドレスと $\overline{\text{RD}}$ および $\overline{\text{MREQ}}$ とがすべてアクティブとなってから、所定の時間以内にデ

図3.6 インターフェース回路のタイミング計算



ータ・バスにデータを出力する必要があります。図 3.8 に示すのはこの計算例です。CPU は Z80A、クロックは 4 MHz でその波形は図に示すように立上り、立下りともに 30ns の対称波形であると仮定します。Z80 のアドレス・バスはバス・サイクルの最初 (図 3.8 の④) から出力されます。ただし、クロックを基準にすると、Z80 の内部回路で最大 110ns の遅れが出ます。この信号が図 3.6 のデコード回路に印加され、一部は 74LS04 を介して、一部は直接に、それぞれ 74LS30 を経て 74LS541 の \overline{G} 入力に加わります。結局、④から 74LS541 の \overline{G} までの最大遅延は 140ns です。一方、 \overline{RD} および \overline{MREQ} は⑤でアクティブとなり、同様に計算すると⑤から 74LS541 の \overline{G} までの最大遅延は 117ns です。74LS541 は 2 本の \overline{G} 入力がともに“L”となってから出力がアクティブとなりますから、結局、⑤からデータ・バスにデータが出力されるまで 155ns が必要です。

一方、Z80 は⑥でデータを読み取ります。この⑥に 50ns のセットアップ時間が必要です。つまり、⑤から 435ns 以内にデータが出力されればよいわけで、先程の値はこれを十分に満足していることがわかります。

なお、以上の計算は図 3.6 の回路を CPU のバスに直接に接続した場合のものです。市販のパーソナル・コンピュータでは CPU のバスにバッファが付加されていることが多く、その場合はバッファの遅延を考慮しなければなりません。また、Z80 の I/O 領域に接続する (I/O マップト) には、 \overline{MERQ} のかわりに \overline{IORQ} を使用すればよく、タイミング的にはいくぶん楽になります。

▶ Z80 以外の CPU につなぐとき 他の CPU を用いた場合の例は第 2 章、2.4 項に示しましたが、さほどの違いはありません。もちろん、バスへの影響やタイミングについては各 CPU に対して計算する必要があります。使用することの多い CPU について主要なタイミング上の数値を計算しておく、回路設計のたびに計算し直さなくて済みます。

例 3-2 ビット単位の入力方法

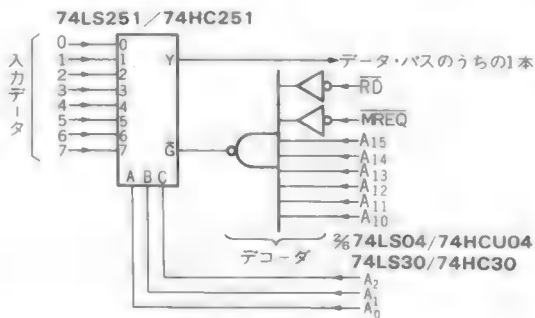
マイコンのデータ・バスは 8～16 ビットの幅であり、アドレス空間を有効に使うには例 3-1 のように多数の信号を並列に取り込んだほうが効率的です。

しかし逆に、少数の信号を 1 ビットずつ取り込んだほうがソフトウェアが楽になる場合もあります。本例ではそのような場合を説明します。

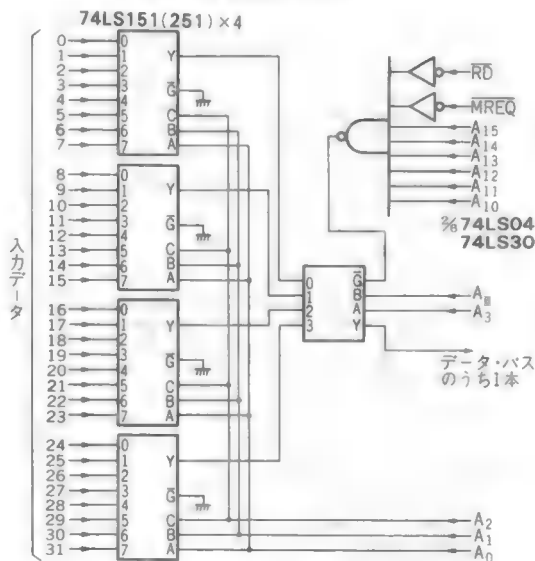
前項で述べた方法は8ビットの信号を1語として並列に取り込むものでした。信号を取り込むこと自体は一度に並列に行うのが簡単なのですが、そのうち特定のビットごとに“1”か“0”かを調べなければならない場合、1つのアドレスで1ビットを取り込んだほうが楽な場合があります。

図3.9に示すのはそのような目的に適した入力用バス・インターフェースの例です。同図(a)は信号が8本以下の場合、同図(b)は8本を超える場合を示します。このような方法で

図3.9 ビット単位の入力用バス・インターフェース



(a) 8ビット以下の例



(b) 9ビット以上の例(32ビット)

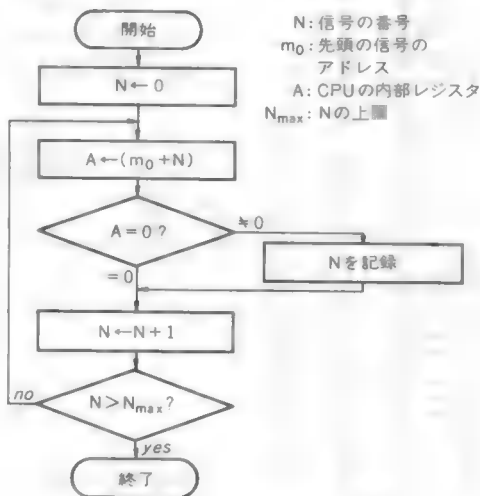
は、データ・バスの幅を1/8しか利用していないわけで、アドレス空間をムダ使いしていることになります。しかし、たとえば「レベルが“1”のビットの番号を記録する」というような使い方に対しては、このほうが便利です(図3.10を参照)。また、データ・バスへの負荷が軽くなる点や、消費電力が少なくてすむ(TTLの場合、出力がスリーステートになるものは許容ファン・アウトが大きいかわりに消費電力も大きい)利点も見逃せません。

例3-3 多数の信号を入力するインターフェース

信号の数が多くなった場合、基本的には例3-1や例3-2の回路を何回路も接続すればよいのですが、いろいろな面で制約を生じます。そこで、多数の信号を接続するためのバス・インターフェース回路の例を示しておきます。

本章の3.1で述べたように、多数の信号を入力しようとする場合は、信号の本数を減らす工夫が必要です。しかし、ソフト上の理由などでやむを得ず信号の本数が増す場合、バスに対する負荷や消費電力を減らすため、ハードウェア上でせめてもの努力をします。

図3.10 “1”の信号の番号を記録する方法



例 3-1 は 1 バイト = 8 本の信号を受けることができます。これを 4 回使えば 32 本の信号を入力できるようになります (図 3.11 (a))。これに対して、マルチプレクサを用いた図 3.11 (b)の回路も、やはり 32 本の信号を入力することができます。

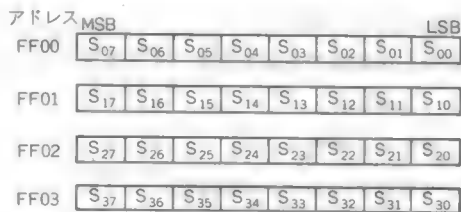
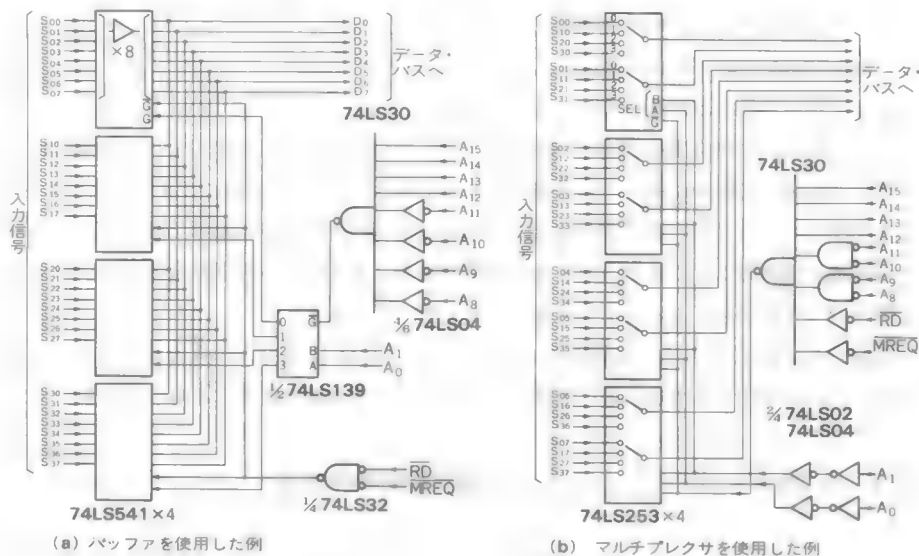
▶バッファを用いるかマルチプレクサを用いるか では、この両者を比較してみましょう。まず、CPU のバスに対する負荷では、

$$(a) I_{OZL}, I_{OZH} = \pm 20\mu A \times 4 = \pm 80\mu A \text{ (max)}$$

$$(b) I_{OZL}, I_{OZH} = \pm 20\mu A \text{ (max)}$$

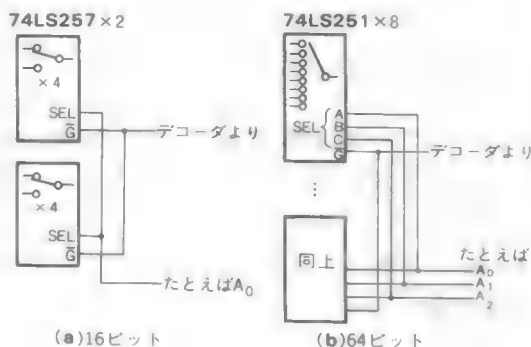
消費電力 (デコード部分を除く) は、

図 3.11 32 本 (4 バイト) の入力用バス・インターフェース



(c) 回路(a),(b)のアドレス・マップ

図 3.12 マルチプレクサを用いた入力用インターフェース



(a) $55\text{mA} \times 4 = 220\text{mA} \text{ (max)}$

(b) $19\text{mA} \times 4 = 76\text{mA} \text{ (max)}$

などと、だいふ違いがあります。

一方、図 3.11 (a)の回路に用いた 74LS541 はバッファ専用の IC です。入力側にヒステリシスがあって雑音に強く、入力電流が少なく ($I_{IL} = -200\mu\text{A} \text{ (max)}$) で通常の LS-TTL の $1/2$) になっていますが、それが必要でない用途には同図(b)のほうが適しています。このへんは些細な問題のようですが、限られた空間や電力、あるいはバス駆動余力をギリギリまで使おうとする場合、このような努力も必要となってきます。

同様にして図 3.12 のような構成にすれば、16 あるいは 64 ビットの入力ができます。もちろん、マルチプレクサを多段に接続すれば、さらに入力数を増すこともできます。

3.3 入力インターフェースの外部機器側の設計

入力インターフェースのマイコン側ポートに TTL を使用した場合、TTL レベルの信号ならば直結できますが、他の信号は接続できません。また TTL レベルであっても変化速度やファン・アウト、雑音などの理由で直結できないこともあります。これら进行处理するのが外部機器側です (図 3.1 を参照)。

この部分はつぎのような要素から成ります。

- (1) 信号の形態やレベルを変換するレベル変換回路 TTL レベル以外のデジタル信号であれば、TTL レベル (あるいはマイコン側ポートが必要とする論理レベル) に変換す

る。また、アナログ信号であればコンパレータ（比較器）やA-D変換器を用いてデジタル信号に変換する。

(2) 信号の変化を吸収するタイミング回路 変化が速すぎてCPUが扱うのに不適当であれば、ラッチ、ワンショット、カウンタなどを用いて適当なタイミングとなるようにする。

(3) 信号の本数を減らすエンコード（符号化）回路 多くの信号を取り扱う場合、簡単な演算で信号の本数が減らせるのであれば、そうしたほうがよい場合がある。また、スキッピングや優先順位についても考慮する。

入力信号の種類は非常に多いため、そのすべてを取りあげることはできませんが、以上の項目を考慮し、代表的なものを以下に示すことにします。

例 3-4 基本的なスイッチの入力方法——チャタリング対策①

マイコンに接続しようとするものの中で、押しボタン・スイッチ、リレー、リミット・スイッチなど、スイッチ類の比重はかなり大きいと思われます。

ところが、スイッチにはチャタリングや接点不良など、特有の問題があります。

本項では、そのような問題をふまえながら、スイッチをバス・インターフェースに接続する方法を説明します。

ここでいうスイッチとは、機械的な接点を示します。したがってリレーは含みますが、オープン・コレクタなどは含みません。

さて、そのような機械的接点の状態を入力する■に大きな問題がいくつかあります。

- (1) チャタリング……水銀接点などの例外を除くと、機械的接点には一度にオンまたはオフするのではなく、開閉時に不安定な状態（開閉を繰り返したり、不安定な抵抗値となる）を通過するが、入力インターフェースでは、これを解決する必要がある。
- (2) スwitchに印加する電圧・電流……マイコンに入力するには、スイッチにある程度の電圧・電流を印加して、スイッチの開閉を電圧レベルに変換しなければならないが、その際の電圧・電流の大きさを適切な範囲にする必要がある。

以上の2点は本来ならば機構部品屋さんの守備範囲なのですが、実用上きわめて重要ですので少し詳しく説明します。

▶チャタリングの時間はどの程度に見積ればよいか これは非常によく質問されますが、答えるほうも一口ではいえないほど範囲の広いものです。しかし、ごく大雑把に言えば、小型のマイクロ・スイッチや小型のトグル・スイッチでは $5\text{ms} \pm 1$ 桁くらいの範囲に入ります。これは、スイッチの製造元にたずねても、あまりははっきりした答は返ってきません。ただし、信号用のリレーなどでは完全に接点が開閉し終わるまでの時間を応答時間として示しているものがあり、チャタリングはそれ以下だということがわかります。

チャタリングに不安をおもひの人は、ぜひチャタリングの実験をしてみることをおすすめします。これは図3.13のような簡単なものでもよくわかります。注意すべき点としてはつぎのようなものがあります。

- (1) 個々のスイッチの間でチャタリング時間には数倍のバラツキがあることもある。少数の実測結果を過信しないこと。
- (2) つき合わせ接点（コラムH参照）のオフの際にもチャタリングが見られる。
- (3) すり合わせ接点では接点が移動している間、相当長い時間にわたってチャタリング（正確に言えば、これはチャタリングではないが）が見られる。

▶接点にはどれくらいの電圧・電流を印加すれば良いか 一般に、大電流用のスイッチを $100\mu\text{A}$ 以下の微小電流で使用しようとしても、接触が不安定で使いものになりません。

図3.13 チャタリングの実験回路

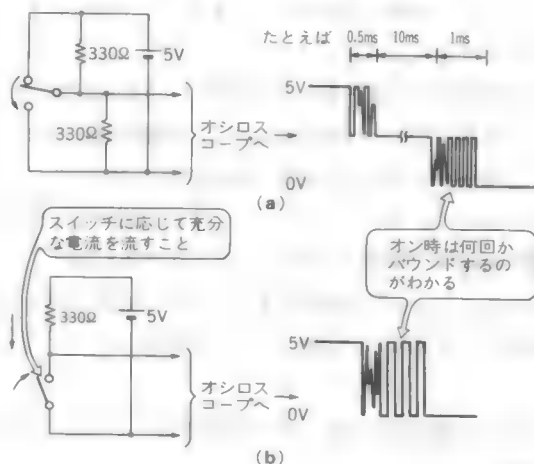
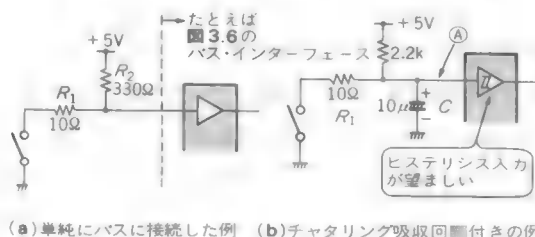


図 3.14 基本的なスイッチ用インターフェース



実際にパネルに取り付けられたり、リミット・スイッチとして用いられているスイッチを見ると、大電流用のトルク・スイッチやマイクロ・スイッチであることが少なくありません。“AC250V 10A”などという定格のスイッチには、少なくとも 10mA 以上の電流を流したいものです。

一方、基板用のスイッチやキー・スイッチで微小電流用のものには、電流を流しすぎると寿命が短くなります。とくに、スイッチに直列に挿入する抵抗(図 3.14 (b)の R_1)を省略したりすると、一時的に多くの電流が流れますので注意が必要です。

▶チャタリングの処理をハード/ソフトのいずれで行うか？ これは■2章でも触れましたが、ハード/ソフトそれぞれの除去能力は一長一短です。しかし、ソフトで行う場合はタイマ割込みでも使わなにかぎり、貴重な時間を浪費していることになります。

また、重要でありながら意外と議論されていないのは、「本当にチャタリングを除去する必要があるのか否か？」という問題です。たとえば、オフの状態のスイッチを監視していて、オンになったら監視ループを抜ければよい場合では、雑音の問題を別にすればスイッチのチャタリングは問題となりません。

▶スイッチ入力インターフェース さて、前置きが長くなりましたがスイッチの入力方法の例を説明します。

図 3.14 (a)は例 3-1 (図 3.6) のバス・インターフェースにたんにスイッチを接続したものです。同一基板内で距離が 50mm くらいであれば $R_1 = 0$ でかまいませんが、スイッチまでの配線を延ばす場合には、 $R_1 = 4.7 \sim 47 \Omega$ を挿入します。これは配線のインダクタンスによる影響を軽減するためです。微小電流用でないスイッチを使用するなら、 R_2 には 10mA 以上の電流を流すようにします。基板用のスイッチには 1mA 以下の電流でも安定なもの

が市販されています。

この回路はチャタリングの除去は全く考慮してありませんし、スイッチの状態も記憶しませんので、必要があればソフトウェアで行います。しかし、ある時点のスイッチの状態がわかりさえすればよい用途も多く、その場合、この回路で充分です。

図 3.14 (b) はよく見かけるチャタリング吸収回路です。小型のマイクロ・スイッチやトル・スイッチの大半はこの回路でチャタリングが除去できます。注意すべき点はつぎのとおりです。

図 3.15 接点に流れる電流による誤動作

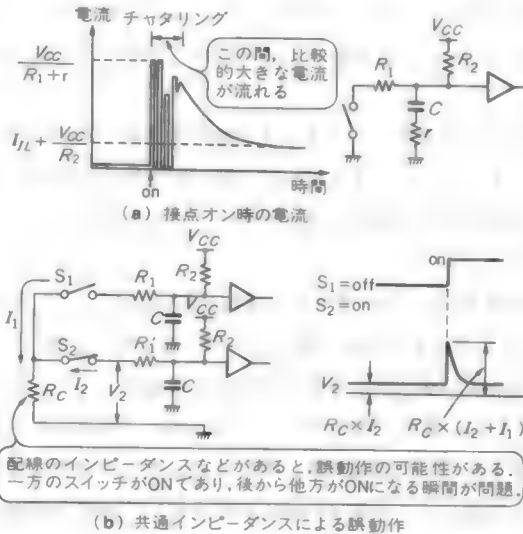
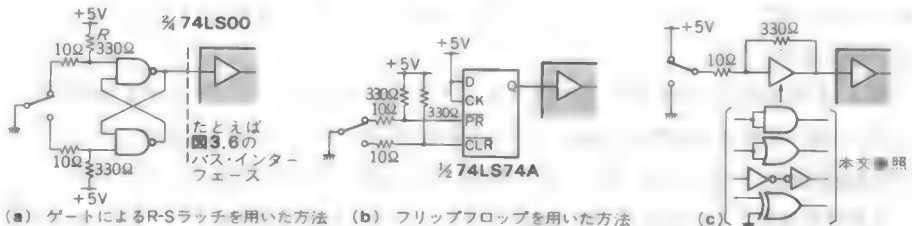


図 3.16 チャタリング除去回路——その 1



- (1) スイッチを閉じた時点で $V_{cc}/(R_1+r)$ の電流が接点に流れる (r はコンデンサ C の内部抵抗および配線の抵抗)。とくに微小電流用のスイッチの定格オーバに注意。また、この電流によって他の回路に誤動作を生じさせないように、配線にも注意すること(図 3.15)。
- (2) 図 3.14 (b) ④ 点の信号は、立上りが遅いので、TTL で受けるのならヒステリシス入力の方がよい (74LS541 はヒステリシス入力)。

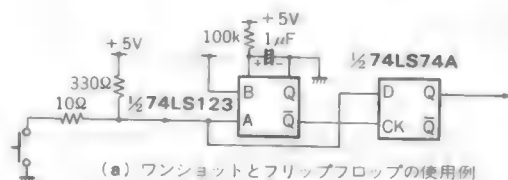
例 3-5 スイッチのチャタリング対策②

スイッチのチャタリングを除去するには他にも多くの方法があります。ここではその中からいくつかを紹介します。

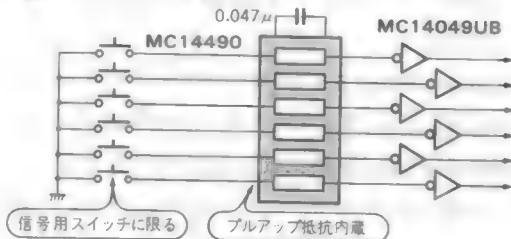
前項でもっとも簡単なチャタリング除去方法を説明しましたので、それ以外の方法について述べておきます。

図 3.16 (a) はトランスファ接点をもつスイッチに適用できるものです。なお、トランスファ接点といっても、必ず BBM (ブレーク・ビフォア・メイク：切換えの瞬間に一度断になる) 動作でなければなりません。これは (b)、(c) の回路でも同様です。

図 3.17 チャタリング除去回路——その 2



(a) ワンショットとフリップフロップの使用例



(b) 専用ICの使用例

図(a)ではゲートでRSラッチを組んでありますが、これを本物のフリップフロップにしても同様です(同図(b))。

また、図(c)はフリップフロップと見てもよいし、極端にヒステリシスの大きいコンパレータと見ることもできます。スイッチに接続する配線が常に低インピーダンスとなり、雑音に強いことが特徴です。ここに使用する非反転のゲートはANDでもORでもかまいません。また、TTLではなくCMOSであっても使えます。

つぎに図3.17に示すのは、ソフトウェアでチャタリングを除くのと同様の考え方で、一定時間のタイマを設けるものです。

同図(a)はワンショット回路とDフリップフロップとを用いたもの、同図(b)は専用IC(MC 14490)を用いたものです。MC 14490の中身は発振器とカウンタとから成る一種のタイマで、IC 1個に6回路が内蔵されています。ただし、TTLは駆動できないのでバッファを必要とします。

例 3-6 多数のスイッチの入力①——エンコード

ロータリ・スイッチやテン・キー、あるいはフル・キーボードのように、スイッチの接点数が多い場合、以上に述べた方法だけでは配線の手間もさることながら、インターフェース回路の量が莫大となってしまいます。

そこでスイッチの種類や使用方法に応じて、入力インターフェースを簡単にするための様々な技法を使用することになります。

ここでは、エンコード(符号化)する方法を説明します。

多数のスイッチがそれぞれ全く独立に開閉するのならばやむを得ません。しかし、本章に示した方法が使える場合、ハードやソフトが簡略化できます。ここではエンコード(符号化)する方法を考えてみます。

エンコードが有効なのは、多数のスイッチのうち閉じているものが1個だけの場合です。エンコードには必然的に優先順位(プライオリティ)が付加されます*ので、2個以上のスイッチが閉じている場合にはそのうち優先順位の高いものに対応する出力が得られます。

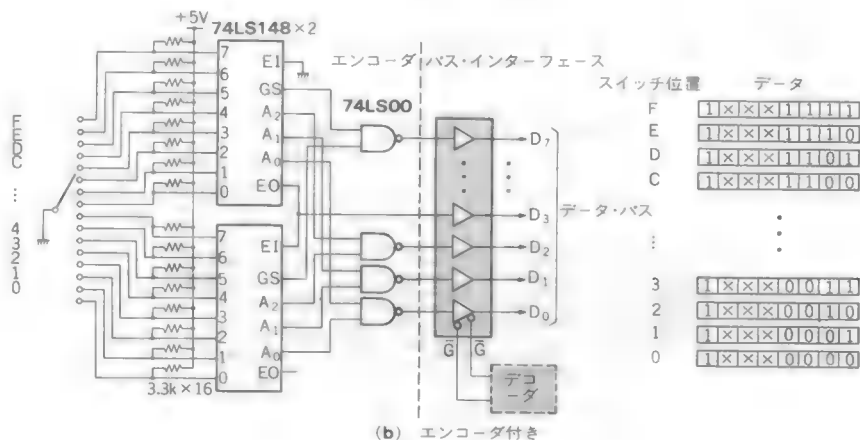
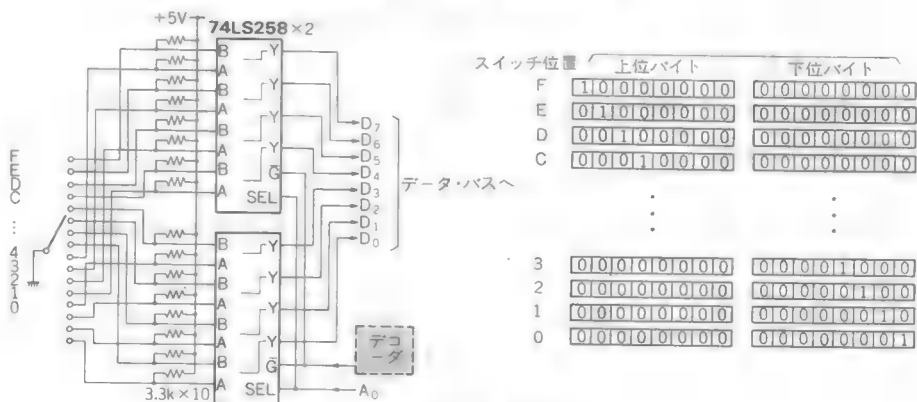
図3.18に示すのは16接点のロータリ・スイッチの入力インターフェースです。同図(a)

* 押した順序まで含めて優先順位をつける(ロール・オーバー)こともある。

はエンコードしないで直接に入力する場合、同図(b)はエンコードした場合です。ハードウェア上の違いは、取り込むビット数が減ることぐらいですが、ソフトウェアでスイッチの接点位置を調べる手間がなくなることも考慮する必要があります。

ロータリ・スイッチの場合、接点位置が同時に2ヵ所以上に渡ることは（過渡状態を除いて）本質的にあり得ないわけですが、図3.19(a)に示すテン・キーなどの場合は生じ得ます。エンコード用のMSIには必ず優先順位がついていますので、テン・キーのうち2個以上のスイッチを押すと、数の大きいほうが出来されるようになります（逆順にすることも

図3.18 ロータリ・スイッチの入力



できる)。

図 3.19 (b) の例は 20 個のスイッチのエンコード用の専用 IC 74C923 を用いた例です。この IC は内部でダイナミック・スキャン(次項を参照)を行っているため、20 個のスイッチは 4×5 のマトリクス状に接続します。なお、74C923 の端子接続を図 3.20 に示します。

エンコード用の IC としては、フル・キー (50 個以上のスイッチから成る) の専用エンコーダもあります。

例 3-7 多数のスイッチの入力②——ダイナミック・スキャン

前項に引き続き、入力インターフェースを簡単にする方法としてダイナミック・スキャンを考えます。

これは、スイッチの組合せ方によって一種のアドレス・デコードをしていると考えれば良いでしょう。

多数のスイッチを入力するには、ダイナミック・スキャンという方法もあります。

図 3.19 キー・スイッチの入力

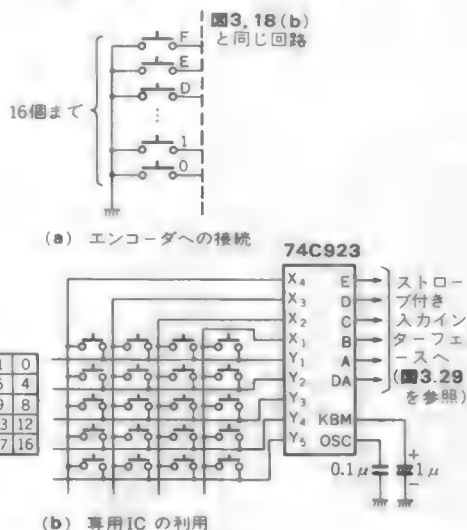


図 3.21 (a)に示すのは複数のロータリ・スイッチの接点位置を調べるもので、アドレスの下位ビットをデコードして1個のスイッチを選択します。図 3.21 (b)は複数のディジタル・スイッチ (サム・ホイール・スイッチ) を読み込むためのものです。ロータリ・スイッチと異なり、ディジタル・スイッチは2カ所以上のポジションが短絡されていることがあるのでダイオードで分離しなければなりません。

その際、ダイオードの順方向電圧によって TTL の $V_{IL} \leq 0.8V$ を満足できなくなることがあるので、CMOS で受けます。また、この回路のようにアドレスの下位ビットをスキャンに使用する場合、必要なとき以外はデコーダ (74LS156) をアクティブにしないほうが

図 3.20 MM 74 C 923 の端子接続⁽¹⁰⁾

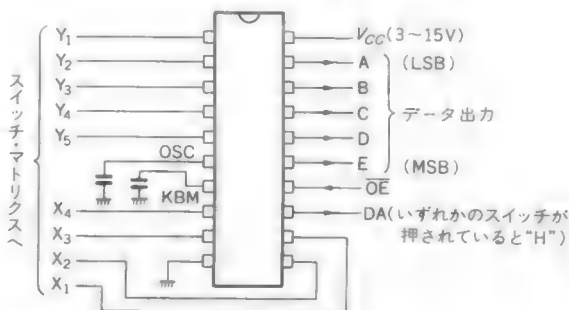
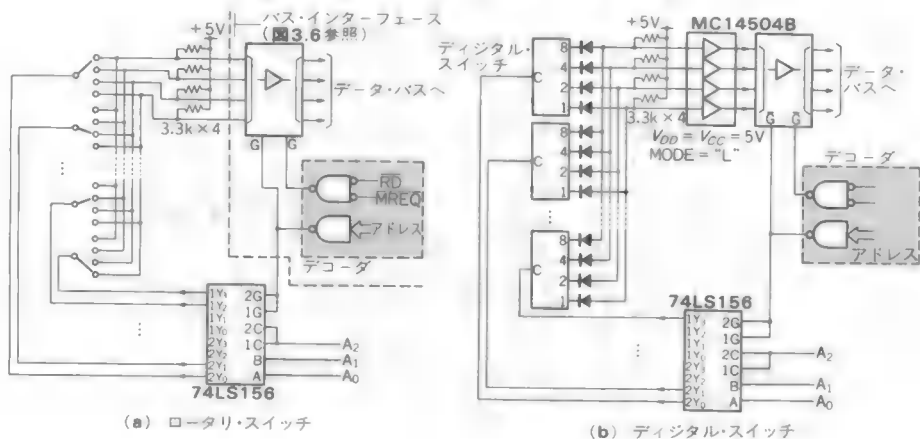


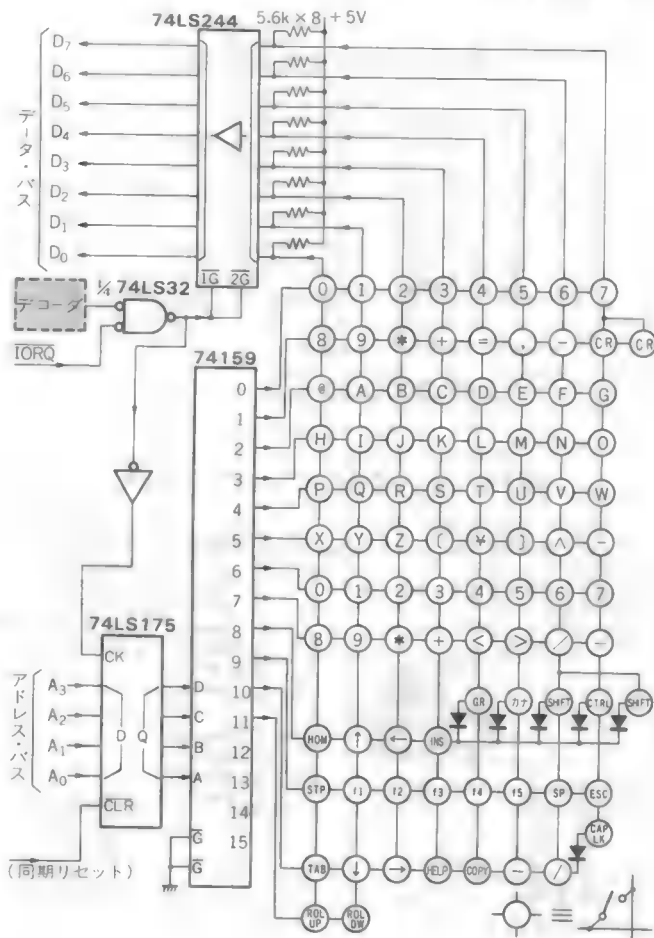
図 3.21 スwitchのダイナミック・スキャン



音などの点で望ましいといえます。

市販のパーソナル・コンピュータのうち価格帯で中以下のものの大部分は、キーボード入りにダイナミック・スキャンを用いています。図3.22は日本電気のパーソナル・コンピュータPC-8801のキーボード入力回路です(日本電気から発表された回路ではない)。どのキーが押されたかの判断はソフトウェアで行っています。なお、このようにダイオードを通った信号をTTLで直接受けるのは、あまり感心できません。

図3.22 パーソナル・コンピュータのキーボード入力方法の例



例 3-8 オープン・コレクタ信号の入力

シーケンサや測定器の出力には、リレー接点と並んでオープン・コレクタがよく使われています。

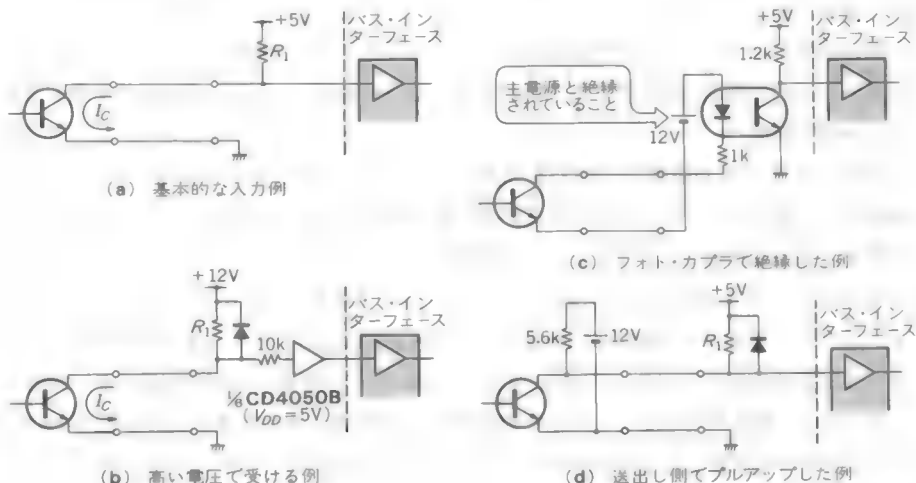
スイッチ類のようなチャタリングや接点不良の問題がない反面、オープン・コレクタはトランジスタとしての電圧・電流上の制約があります。

本項ではオープン・コレクタ信号をバス・インターフェースに接続する方法を説明します。

オープン・コレクタ (FET 出力ならばオープン・ドレイン) 出力はチャタリングの問題はなく、しかも接点と同様に任意の電圧レベルに変換容易な利点があり、またフォト・カプラを使えばアイソレーション (絶縁) が可能なこともあって、デジタル信号の受け渡しに広く使われています。

一方、オープン・コレクタは接点と違って飽和電圧が存在し、オフ時にも漏れ電流があります。また、耐圧やコレクタ電流の制限もあるので注意が必要です。

図 3.23 オープン・コレクタ信号の入力



▶オープン・コレクタの入力回路 図3.23(a)に示すのはオープン・コレクタ信号の基本的な入力方法です。ダーリントン接続でないオープン・コレクタ出力は、オン時の飽和電圧は0.3V程度以下になりますから、TTLやCMOSで直接受けることができます。プルアップ抵抗（オープン・コレクタの負荷の一部）はオフ時の漏れ電流から決定します。たとえば図(a)のようにTTLで受ける場合 R_1 はつぎのように選びます。

$$\frac{V_{CC}(\max)}{I_C(\max)} \leq R_1 \leq \frac{V_{CC}(\min) - V_{IH}(\min)}{I_{IH}(\max) + I_{CE}(\text{OFF})}$$

$$\left[\begin{array}{l} V_{CC} : \text{電源電圧 (5V} \pm 5\%) \\ V_{IH}(\min) : \text{TTLの“H”レベル電圧の下限 (2.4V)} \\ I_{IH}(\max) : \text{TTLの“H”レベル流れ込み電流の最大値 (40}\mu\text{A)} \\ I_{CE}(\text{OFF}) : \text{オープン・コレクタのオフ時の漏れ電流} \\ I_C(\max) : \text{オープン・コレクタの最大許容コレクタ電流} \end{array} \right]$$

たとえば $I_{CE}(\text{OFF}) = 100\mu\text{A}$ 、 $I_C(\max) = 10\text{mA}$ とすると、 $525\Omega \leq R_1 \leq 16\text{k}\Omega$ となります。一般にオープン・コレクタが使用されるのは雑音環境のあまりよくない場所なので、 R_1 は低目にしたほうが安全です。

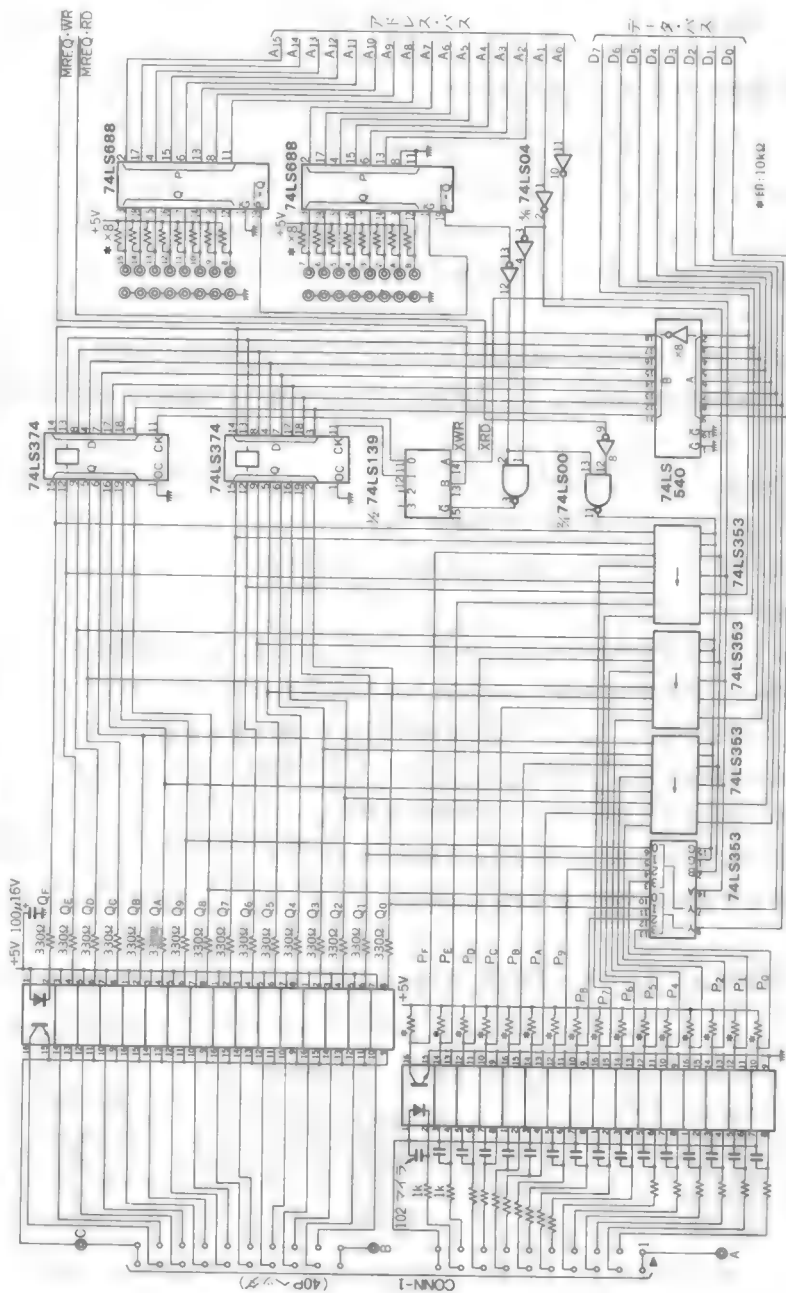
図3.23(b)はオープン・コレクタを高い電圧で受けるものです。これは耐雑音性などからレベルを高くしておきたい場合のほか、オープン・コレクタ側がダーリントン接続になっている場合（オン時の飽和電圧が0.7V以上になるため、TTLでは直接は受けられない）にも使えます。

同図(c)はオープン・コレクタ信号をさらにフォト・カプラで絶縁した例です。雑音環境が悪く、しかも何をつながれるか予測できないような場合、とにかく電氣的に分離しておくのが安全です。

図(d)は送出し側の装置内で他の電源電圧にプルアップされている場合です。こういう接続はよくあるのですが、このままでは高い電圧が入力に印加されることになるので、受け側で正規の負荷抵抗とダイオードとを接続します。

図3.24に示すのは産業用の入出力インターフェース基板で、入出力各16ビットを持ち、すべてフォト・カプラで絶縁しています。この回路では出力したデータを読み戻すことができるようになっています。入力用のフォト・カプラのLED側に並列に接続されているコンデンサは、雑音対策用です。マルチプレクサ(74LS353)のセレクト入力(A, B)にはアドレス信号を反転したものが接続されており、入力の番号とCPUから見たアドレスと逆順になっていることに注意して下さい。なお、出力インターフェース(例4-1, 例4-9な

図 3.24 入出力インターフェース基板



ど) もご参照下さい。

例 3-9 他のロジック・レベルの入力

接点でもオープン・コレクタでも TTL レベルでもない信号は、何らかのレベル変換を行ってからバス・インターフェースに接続する必要があります。

本項では、そのなかのいくつかを紹介します。

他のロジック・レベルとは、電源電圧 5V の TTL や CMOS あるいは接点などと異なるデジタル信号を指しますが、多くの種類がありますので、よく使うものから例をあげてみます。

- (1) CMOS レベル (5~15V)
- (2) リレー・シーケンサなどの制御機器 (24V)
- (3) RS-232 C/RS-423 など正負に振る電圧レベル ($\pm 3 \sim \pm 15V$)
- (4) 電流ループ (20mA, 60mA)

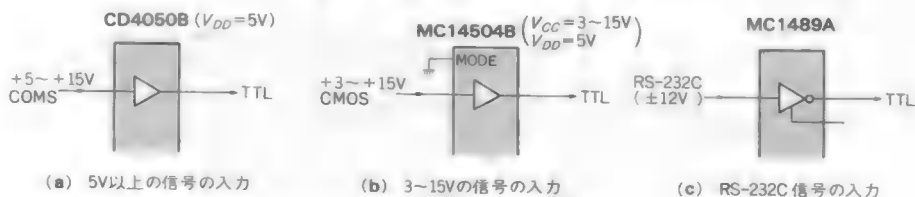
このような多様なレベルとのインターフェースを行うとき、もっとも簡単かつ確実なのは専用のレベル変換用 IC を使うことです。たとえば、図 3.25 (a), (b) は +15V までの CMOS レベル、(c) は RS-232 C ($\pm 12V$) の各変換用 IC です (第 5 章も参照)。

もちろん、個別部品や一般的な IC を用いてレベル変換を行うこともできます。図 3.26 に示すのは個別部品を使ったレベル変換回路の例です。

図 3.27 (a) に示すのはノートン・アンプを用いたものです。ノートン・アンプ (電流入力の単一電源用増幅器) の出力は TTL と相性がいいので、もっと多く使われてもよいと思います。

図 3.27 (b) はコンパレータ (比較器) を用いたもの、(c) はフォト・カプラを利用したもの

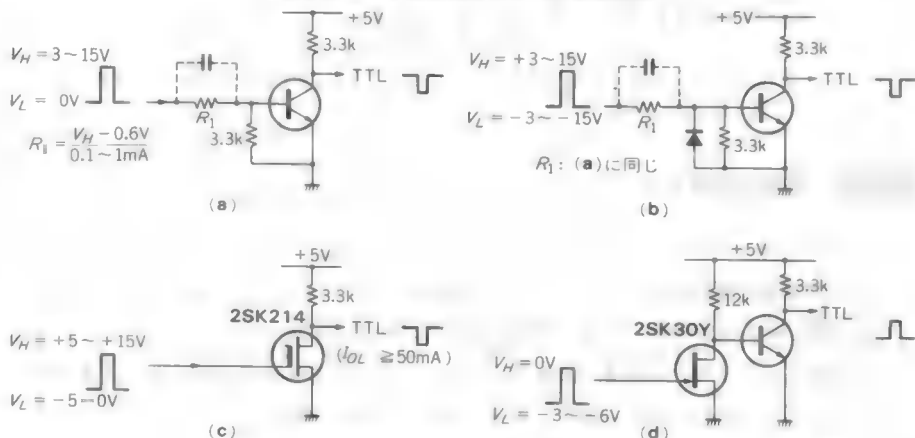
図 3.25 各種のレベル変換回路



です。フォト・カプラは絶縁とともにレベル変換が可能です。同図(d)は CMOS のコンパレータですが、入力電圧を（電源電圧 - 2V）以下に抑えねばなりません。

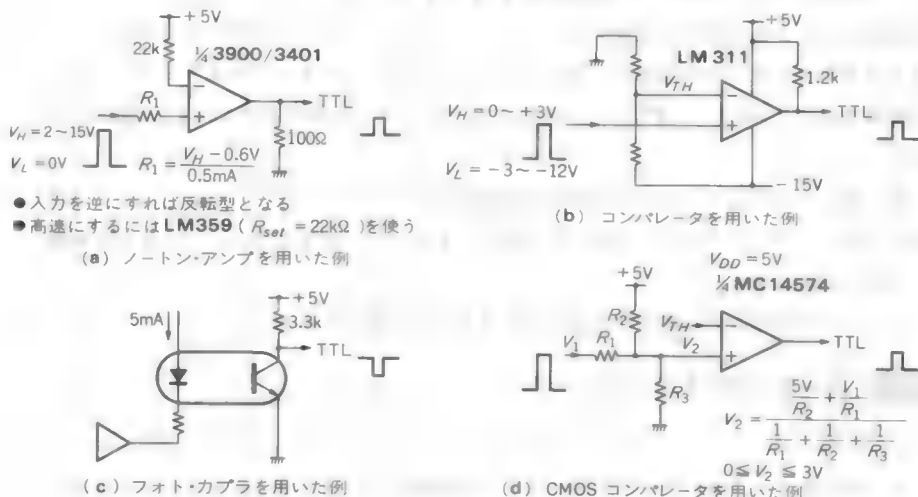
一般に、専用のレベル変換用 IC に比べて、個別部品などで組んだレベル変換回路は速度が遅くなりがちなので注意が必要です。

図 3.26 個別部品によるレベル変換回路



注：トランジスタはnpn 小信号用(2SC2785など)

図 3.27 その他のレベル変換回路



- 入力を逆にすれば反転型となる
- 高速にするには LM359 ($R_{set} = 22\text{k}\Omega$) を使う

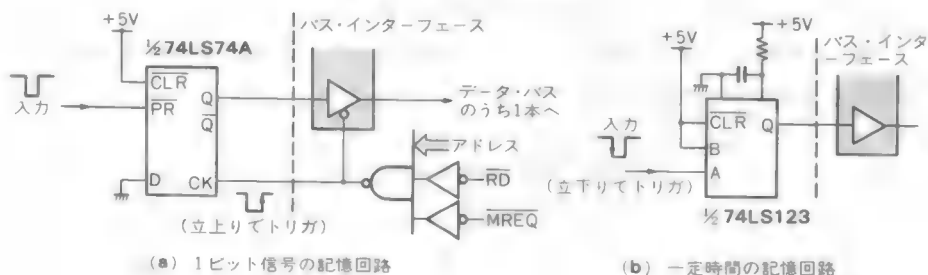
(a) ノートン・アンプを用いた例

(b) コンパレータを用いた例

(c) フォト・カプラを用いた例

(d) CMOS コンパレータを用いた例

図 3.28 状態を記憶する



例 3-10 状態を記憶する

信号の変化が速い場合、CPU が信号の変化を読みとれないことがあります。そのような場合、インターフェース側で信号を保存することが必要です。

この中にはストロープ付きの信号やハンドシェイクなども含まれますが、まず本項では“スイッチが押されたか否か”のような簡単な場合を例にとります。

たとえば、信号が1回印加されたら、つぎにそれを読み込むまで記憶しておかねばならないことがあります。そのような場合は図3.28のような方法で状態を記憶します。

同図(a)は1ビットの信号を記憶するものです。1回でも信号が印加されると状態が“1”になりますので、読み込んだあと、何らかの方法でリセットする必要があります。リセットの簡単な方法としては、CPU からの書き込みストロープの使用があげられます(例4-6を参照)。

一方、ポーリング・ループを回る時間だけ状態を保てばよいのなら、ワンショット(単安定マルチバイブレータ)でも可能です(図(b))。もちろん、時定数はループをまわる時間よりも長く設定します。

ただし、いずれの方法も雑音には十分に注意を払う必要があります。

例 3-11 ストロープ付きの入力

キーボード・エンコードなどでは、スイッチが押されたタイミングを示すとともに、

データが有効となったことを示すためストロブ信号が出力されます。また、測定器などでは信号を1桁分ずつ時分割で出力するとともに、それが何桁目のデータかを示すストロブ信号が付加されていることがあります。

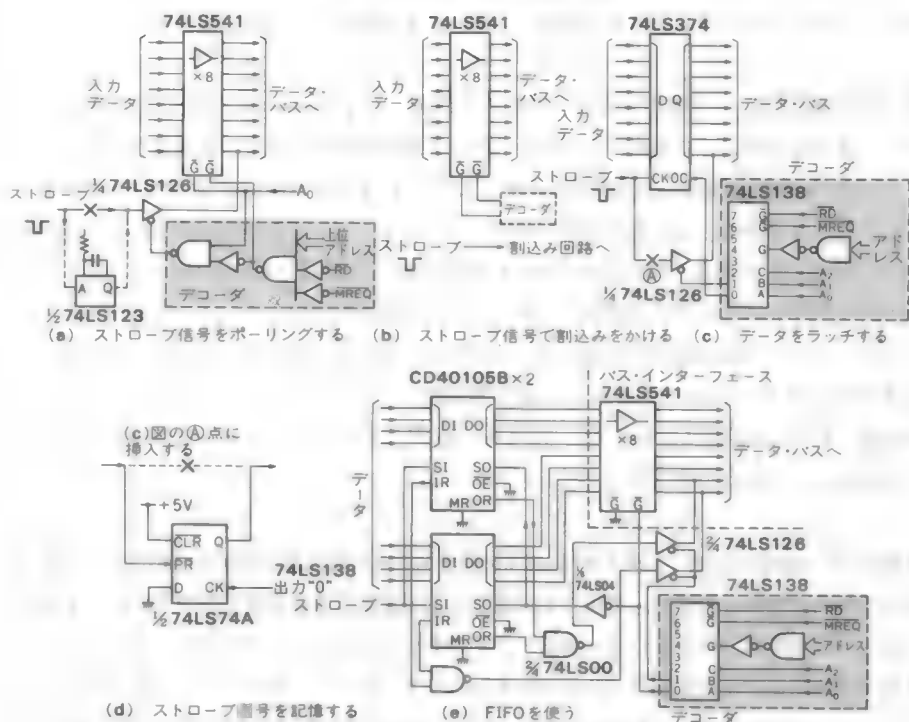
本項では、このようなストロブ信号の使い方を示します。

▶ **キーボード回路** エンコーダ付きのキーボードなどでは、キーを押すとエンコードされた値とともに、データが有効であることを示すストロブ信号が出力されます。

例3-6の図3-18(b)に示したエンコード回路では、74LS148のGS出力がこのストロブ信号に相当しますが、これは1個以上のスイッチを押している間はずっとアクティブになったままです。

これに対して、フル・キーボードなどに使われているエンコーダは、スイッチを押した

図 3.29 ストロブ付きの入力



あと所定の幅のパルス状のストローブ信号を出力します。このパルス幅は1~100 μ sのものが多いようです。

▶ストローブ入力信号の処理方法 ① さて、マイコン側がストローブ信号のポーリングに専念できるならば、100 μ sのストローブ・パルス■を読み取ることは可能です。ストローブ・パルス幅が短かければ、ワンショットで100 μ s程度まで延ばす手があります(図3.29(a))。

マイコンが十分な速さでのポーリングを行えない場合、もっとも有効なのはストローブ信号で割込みをかけることです(図(b))。ただし、この方法も図(a)に示した方法も、ストローブ信号がアクティブでなくなってからもデータが保持される場合にしか使用できません。

確実な動作を望むのなら、図3.29(c)、(d)のような回路を使用します。いずれもストローブ信号を用いてデータをラッチするものですが、(c)は充分速いポーリングまたは割込みを使用するもの、(d)はそうではない場合に使用できるものです。すなわち(d)の回路は、ストローブ信号が印加されたことを記憶し、データを読み出すとともに戻ります。

▶FIFOの使用法 以上にあげたいずれの方式も、マイコンがデータを読み取らないうちにつぎの信号が入った場合、古いほうのデータは何の断わりもなく更新されてしまいます。これを避ける方法の一つはFIFOを使うことです。FIFOは印加されたデータを順送りに吐き出しますから、一時的に読取りが間に合わなくてもデータは失われません。図(e)に示すCD40105Bの場合、FIFO内にデータが残っていればOR(Output Ready)端子が“H”になりますから、このフラグを調べてからデータを読み出せばよいことになります。

なお、万一、FIFOが満杯のときにさらに新たな、データが印加された場合、そのデータはFIFOには入りません。このような事態が生じたことを検出するには、IR(Input Ready: FIFOが満杯になると“L”になる)が“L”のときにストローブが印加されたことを記憶しておけば可能です。

▶BCDデータの入力法 図3.30に示すのはデジタル電圧計⁽¹³⁾からのBCDデータをマイコンに入力するものです。デジタル電圧計などでは信号を多重化したうえ、ストローブ信号で区別することがよく行われます。ラッチが2段になっているのは、4桁分のデータ出力が終了して1回分のデータとみなすためです。ただし、後段にデータをラッチしようとしている時点でデータを読み出せば、やはりその内容は保証されませんから、CPUが

らのデータ読出しは必ずステータス（図 3.30 では EOC 信号）を調べてからにします。

例 3-12 ハンドシェイク入力

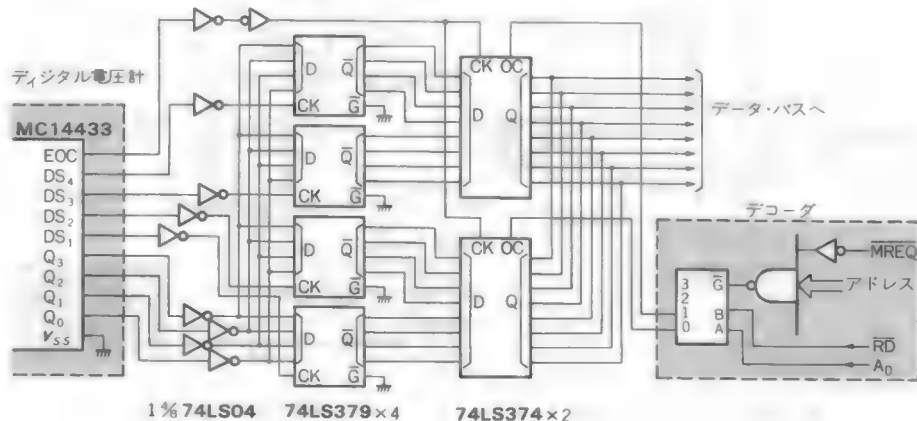
確実な信号転送を行うには、送り出し側が次の信号を出力する前に、受け取り側が前の信号を入力したのを確かめる必要があります。この確認操作がハンドシェイクです。

ハンドシェイクはプリンタとのインターフェース（セントロニクス型ハンドシェイク）や GPIB など、マイコンと周辺機器とのインターフェースに幅広く使われています。なお、第 4 章のコラム J も参照してください。

データ転送の際に、転送が確実に行われたことを確認しながら行うには、ハンドシェイクを用います。

図 3.31 に示すのは基本的なハンドシェイクの構成で、送り出し側がデータを出力するとともに、フリップフロップをセットします。受け取り側はフリップフロップの状態を調べて、セットされていればデータを読み込むとともにフリップフロップをリセットします。送り出し側が新たにデータを出力する場合、フリップフロップがリセットされていることを確認してから行います。このようなハンドシェイクを行えば、データが消失することはありません。

図 3.30 デジタル電圧計からのデータ読込み



▶ハンドシェイクの回路例 図 3.32 に示すのはハンドシェイク型入力インターフェースの例です。

図(a)は基本的なもので、ストローブ信号でフリップフロップをセットし、読出しと同時にリセットします。このような方式を2線式ハンドシェイクと呼びます。なお、データの送り出し側がハンドシェイク終了までデータを保持している場合はデータをラッチする必

図 3.31 基本的なハンドシェイク

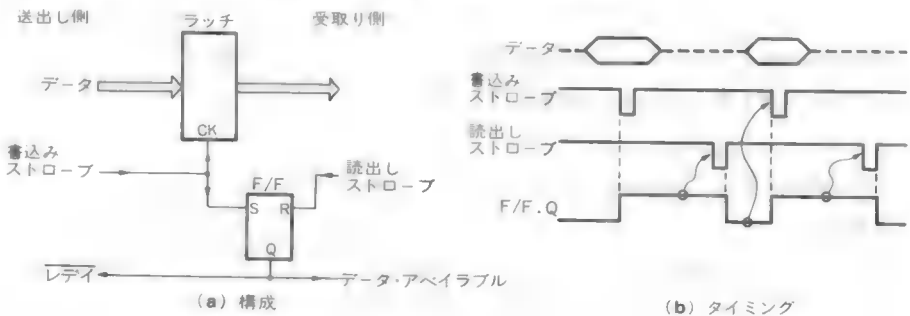
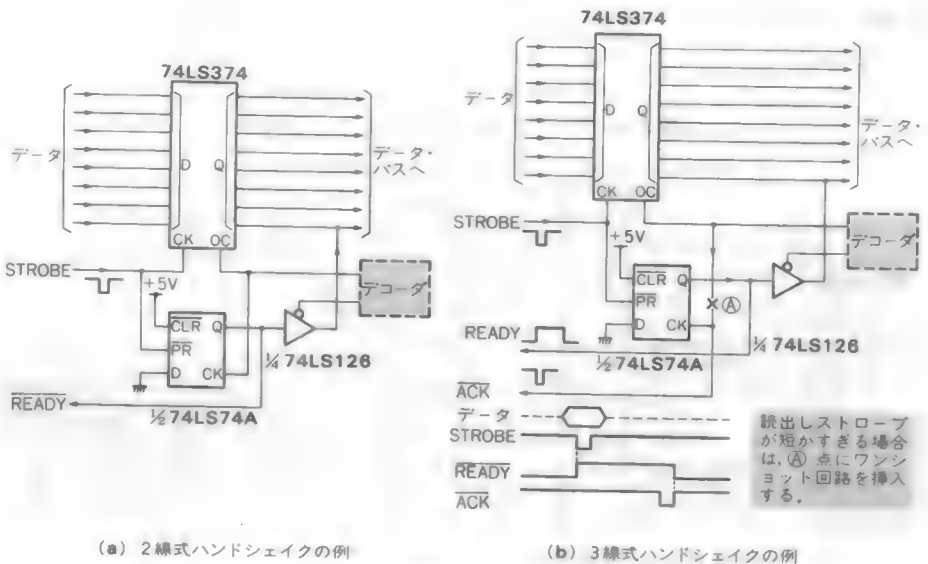


図 3.32 ハンドシェイクを用いた入力



要はなく、74LS374 のかわりに 74LS541 などのスリーステート・バッファでも使えます。

図(b)は 3 線式ハンドシェイクです。GPIO (IEEE-488) と呼ばれるインターフェースは、基本的にはこの 3 線式ハンドシェイクです。

メーカ製のパーソナル・コンピュータにはプリンタ用の出力ポート（多くはセントロニクス型と称している）が付いており、BASIC の PRINT 文で簡単にデータを出力できます。これに対して図 3.31、図 3.32 のような入力インターフェースをもっていれば、他のマイコンからのデータをもっとも簡単に受け取ることができます。

なお、ハンドシェイク出力の方法については例 4-7 も参照してください。

例 3-13 基本的なアナログ入力①——レベル検出

マイコンにアナログ信号を入力する場合、一般的には A-D 変換器を使用しますが、“所定のレベルを越えたか否か”を知りたい場合はレベル・コンパレータ（比較器：1 ビットの A-D 変換器と見てもよい）が使用できます。

ここではその例と、基本的な注意事項とを示します。

アナログ信号といっても多くの種類がありますが、ここでは、信号のレベルや形態は A-D 変換器の入力に適合するように処理されているものとしします。

さて、アナログ信号をマイコンに取り込むには A-D 変換を行わねばなりません。アナログ入力インターフェースの設計の中では、A-D 変換器を選択することがもっとも重要だといえるでしょう。

従来、A-D 変換器は高速の逐次比較型と高分解能の 2 重積分型とが代表的といわれてきました。しかし、最近になって従来の欠点を改良した高速・高分解能の A-D 変換器が数多く発表されています。また、マイコンが必要とする A-D 変換器は、従来の計測用のものに比べて速度・分解能ともに多少異なったものとなっています。したがって、本書で述べる A-D 変換器の分類は従来のものとは必ずしも一致しません。

▶ **レベル検出回路** 図 3.33 に示すのは簡単なレベル検出回路です。A-D 変換器のもっとも簡単なものは、このようなコンパレータ（比較器）です。レベル検出回路の設計には、つぎのような点に注意します。

(1) 入力レベルに注意する。たとえば LM319 は、+5 V（単一）にまで電源電圧を下げ

で使うことができ、デジタル回路との共存に便利だが、+5V で使ったときの入力電圧範囲は+1～+3V に過ぎない。

(2) 必要に応じてヒステリシス（正帰還）を考慮する。これは遅い信号変化の際のマルチプル・トリガを防止するためと考えたらい、上側・下側の二つの離れたしきい値をもたせたいのであれば、コンパレータを2個使ったほうが簡単である。

▶ **応答速度の問題** このほかコンパレータの応答速度の問題がありますが、2901 系のような比較的低速のものでも $1 \sim 2 \mu\text{s}$ で、ソフトウェアで読み込む速度に比べれば、1桁は高速です。

さらに高速なコンパレータが必要ならば、図 3.34 に示す $\mu\text{A}760$ (25ns (max)) のような高速のものを用品。また、SN75207 などのライン・レシーバやセンス・アンプは一般に高速で、しかも出力が TTL レベルとなっているので意外に便利です。欠点は、入力バイアス電流やオフセット電圧が本物(?)のコンパレータよりも大きいことです。

図 3.33 コンパレータを用いたレベル検出

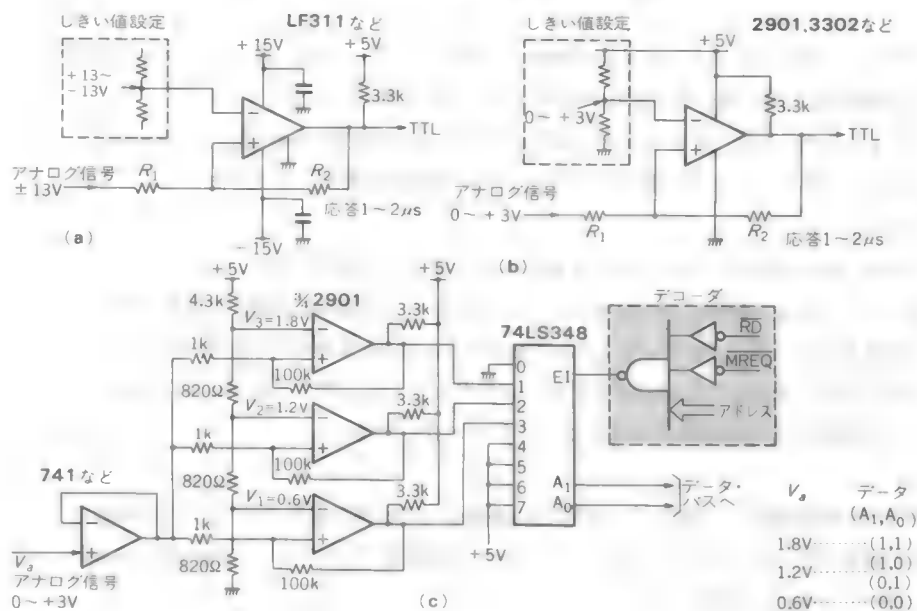
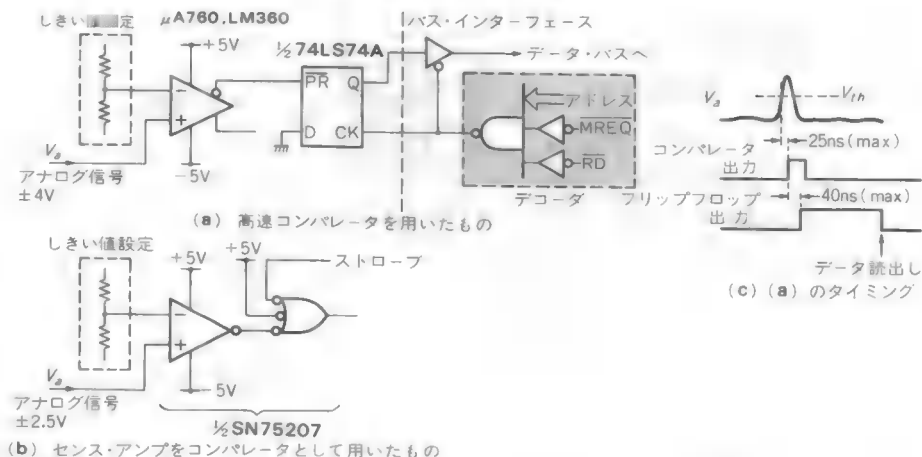


図 3.34 高速のパルス検出



例 3-14 基本的なアナログ入力②——8ビット中速A-D変換器

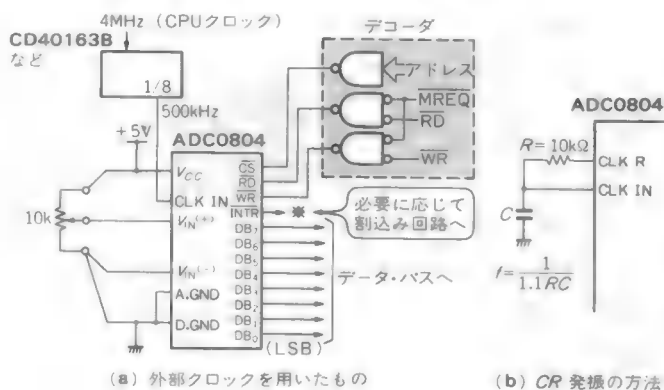
アナログ信号を1%程度の精度や分解能で取り込みたい場合、8ビットのA-D変換器が便利で、これはマイコンのデータ・バスに一致しており、処理も簡単です。本項では、このようなA-D変換器の使い方を紹介します。

センサやトランスジューサからの簡単な信号の取込み、ボリュームやポテンショメータの位置の読み込みなどのうち8ビット（1バイト）以下の分解能ですむのであれば、安価なA-D変換器が数多くあります。

図3.35に示すのはNS社の8ビットA-D変換器ADC0801シリーズ⁽⁸⁾です。このICはセカンド・ソースも多く、小型で安価です。このICのA-D変換方式は逐次比較方式なのですが、従来使われてきたR-2Rラダーではなく、リニアな分圧回路を用い、内部のコンパレータもダイナミック型とするなど、新しい技術が用いられています。ディジタル部分はCMOS構造として消費電力を抑えています。

■ A-D変換回路例 図3-35(a)は5Vの電源に接続されたポテンショメータの位置を読み込む例です。このICの入力電圧範囲は原則として0～5V（電源電圧以内）であり、し

図 3.35 8ビット中速のA-D変換



かも電源電圧を基準電圧源として使用します。したがってポテンショメータのように電源電圧に対する分圧比を求めるには最適といえます。

変換にはクロック信号が必要で、同図(a)のようにマイコン側のクロックを流用してもよいし、(b)のようにCRで発振させることもできます。外部からクロックを印加する場合、デューティ比が50%(±10%)となるように注意します。クロック周波数と変換時間との関係は、

$$\text{変換時間} = (1/\text{クロック周波数}) \times (66 \sim 73)$$

となります。精度を悪くしない範囲では、クロック周波数 640kHz、変換時間約 110μs が上限です。

▶マイコン・バス側 つぎにマイコン・バス側を見てみましょう。このICは $\overline{\text{CS}}$ と $\overline{\text{WR}}$ とをとともに“L”にすると、そのいずれかの立上りから変換動作を開始します。このときのデータ・バスの信号は何でもかまいません。変換を開始すると $\overline{\text{INTR}}$ 端子が“H”になり、終了すると“L”になりますから、この状態を割込みやポーリングで検出するか、あるいは必要な変換時間だけ待機して、データを読み出します。読出しは $\overline{\text{CS}}$ と $\overline{\text{RD}}$ とをとともに“L”にします。アクセス時間は最大で 200ns ($C_L=100\text{pF}$) ですから、4MHzのZ80Aにはメモリ領域、I/O領域のいずれにも直結できます(図2.9参照)。直流的には $V_{OL}=0.4\text{V}$ ($I_{OL}=1.6\text{mA}$) です。

なお、このA-D変換器のアナログ入力端子のインピーダンスは内部のスイッチングに

より変動しますので、ポテンショメータまでの距離が長い場合は注意が必要です。

例 3-15 高速のアナログ入力——高速 A-D 変換上の問題点

アナログ信号もその変化が高速になるとマイコンでの処理はしだいに困難になります。そのため信号処理専用のハードウェアも出てきています。しかし、音声帯域までならばマイコンでも何とか取り扱えます。

本項で説明するのは変化するアナログ信号をマイコンで入力する際の基本についてです。

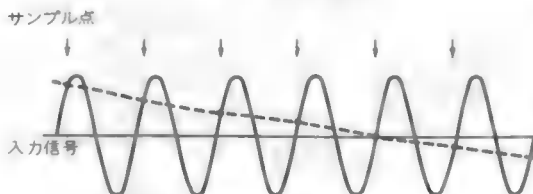
高速といっても、ここでは変換周期 $30 \sim 300 \mu\text{s}$ くらいを考えます。これ以上の速度になるとソフトウェアによる読込みは困難で、DMA を考慮せざるを得ません。この速度であってもマイコン側は一定数のデータを取り込む間はそれに専念することが必要です。

一般に、短い変換周期を必要とするのは、入力信号もそれに応じた変化をともしう場合です。ところが、積分型は別として、逐次比較型などの A-D 変換器は、原則として変換中に入力信号を変化させると、正しい A-D 変換ができません。また、変換周期に近い周波数の信号を入力すると、折返し歪が現れます (図 3.36)。

▶ローパス・フィルタとサンプル・ホールド回路 このようなことから、高速の A-D 変換を行うには入力信号の帯域に応じて、ローパス・フィルタやサンプル・アンド・ホールド回路を付加する必要があります。

まず、ローパス・フィルタについて考えます。A-D 変換器のようなサンプル値系では標

図 3.36 折返し歪の発生



実線の入力信号と点線の入力信号との区別がつけられなくなる。

例 3-16 高分解能アナログ入力——必要な精度と分解能を得る方法

工業用の測定では遅くともよいかわりに 10 ビット以上の分解能が必要なことがあります。

しかし、高い分解能や精度が必要な場合、たんに所定ビット数の A-D 変換器を接続するだけでなく、いくつかの注意が必要となります。本項では、そのような場合の基礎事項を説明します。

工業計測用としては 10 ビット以上の A-D 変換が必要とされることがあります。ビット数と分解能との関係を表 3.1 に示しますが、符号込みの 12 ビットはだいたい $3\frac{1}{2}$ 桁に相当します。

▶**分解能と精度** ここでひとつ注意しておきますが、一般に分解能と精度（広義の）とは異なります。つまり、分解能 12 ビットで精度（たとえば直線性誤差）0.1% ということはありません。

また、A-D 変換器には必ず量子化誤差（ $\pm 1/2$ ビットに相当する）が存在します（図 3.

表 3.1 分解能・ビット数と量子化誤差

分 解 能 (符号込み)	フルスケールのディジタル値		量 子 化 誤 差
	符号なし	符号あり	
6 ビット	0～ 63	-32～+31	0.78 %
7	0～ 127	-64～+63	0.39 %
8	0～ 255	-128～+127	0.20 %
9	0～ 511	-256～+255	0.10 %
10	0～ 1023	-512～+511	0.049 %
11	0～ 2047	-1024～+1023	0.024 %
12	0～ 4095	-2048～+2047	0.012 %
13	0～ 8191	-4096～+4095	61 ppm
14	0～16383	-8192～+8191	31 ppm
15	0～32767	-16384～+16383	15 ppm
16	0～65535	-32768～+32767	7.6ppm

38(a)). A-D変換器の誤差の表記にはいろいろな方法がありますが、全体の精度を求めるには表記されている誤差のほかに量子化誤差を考えることを忘れないようにします。

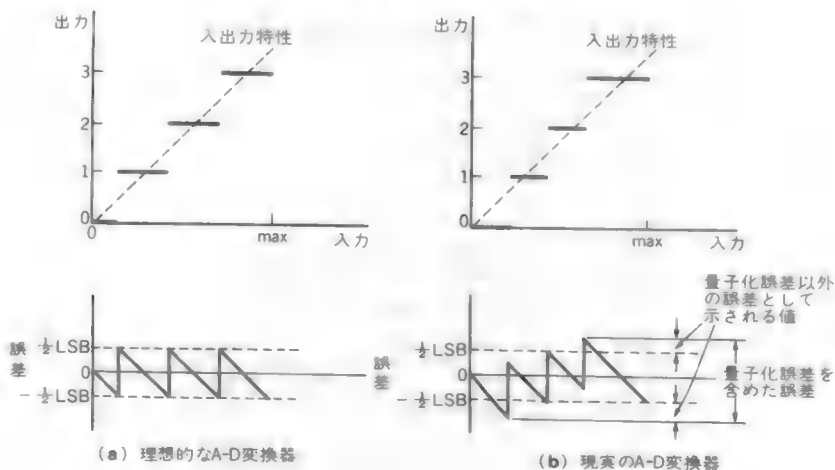
▶高分解能A-D変換回路 さて、10ビット以上の分解能を必要とするアナログ信号の入力方法を図3.39に示します。

図3.39(a)はインターシル社のICL7109を用いたものです。このA-D変換器は極性+12ビットの分解能をもち、入力範囲は $\pm 0.4\text{V}$ （分解能 $10\mu\text{V}/\text{ビット}$ ）以上で任意に設定でき、出力がスリーステートになるなど、マイコン用として多くの利点をもっています。A-D変換は基本的には2重積分型で、毎秒30回くらいが限界ですので低速の部類に入りますが、マイコンを用いた工業用計測には幅広く使えると思います。

このA-D変換器のRUN/ $\overline{\text{HOLD}}$ 端子は“H”にすれば変換を開始し、“L”にすれば続行中の変換を終了してから停止します。“H”に固定しておけば連続して変換を続けます。

変換が終わるとSTATUS端子が“H”から“L”になりますので、これを割込みまたはポーリングで検査してデータを読み込みます。出力は図3.39(b)のようなビット構成となっています。ICL7109自体もバス・インターフェースを内蔵しており、そのアクセス時間は $\overline{\text{CE}}$ から400ns、 $\overline{\text{HBEN}}$ または $\overline{\text{LEBN}}$ からは各350nsとなっているので4MHzのZ80AのI/O領域への直結も可能です（図3.39(c)）。ただし、バス負荷が増した場合の遅れの度

図3.38 A-D変換器の誤差



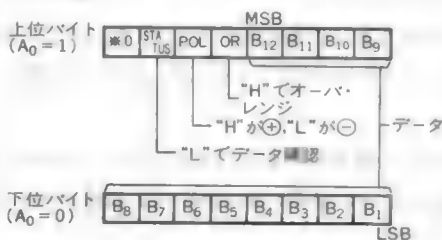
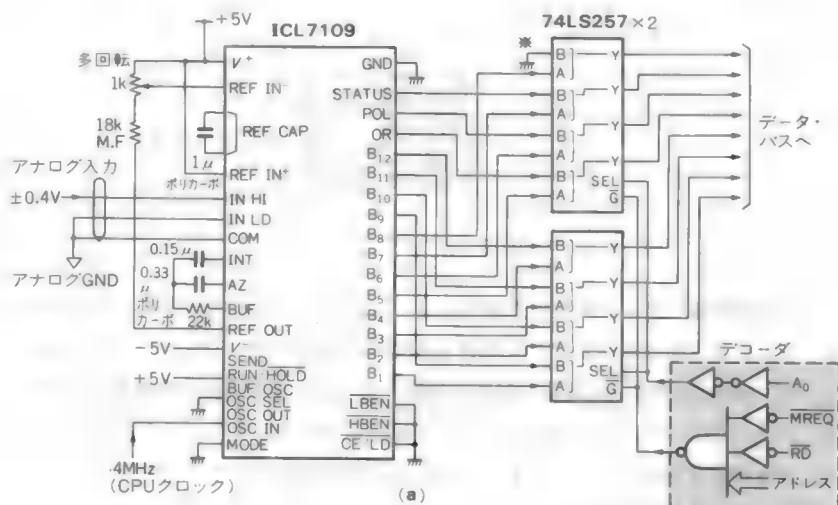
合いが明示されていない点と、出力が非アクティブ(スリーステート)に戻るまでの時間が長いバス衝突の可能性がある点とを考えると、バッファを付加したほうが安全です。

▶ 必要な精度を得るためには このように分解能が12ビット以上であっても、マイコンに接続すること自体は簡単です。ただし必要な精度を得ることは決して簡単ではありません。詳しいことは他書にゆずるとして、ここでは二つの項目を示しておきます。

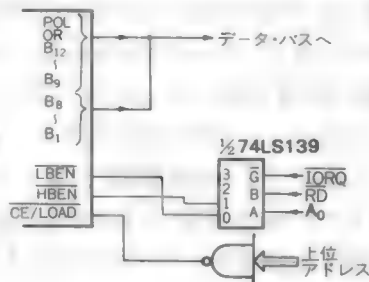
(1) アナログ・グラウンドとデジタル・グラウンドとは分離されているか。

マイコン自体は巨大なデジタル回路であり、発生する雑音は決して少なくありませ

図 3.39 12ビット低速 A-D 変換器

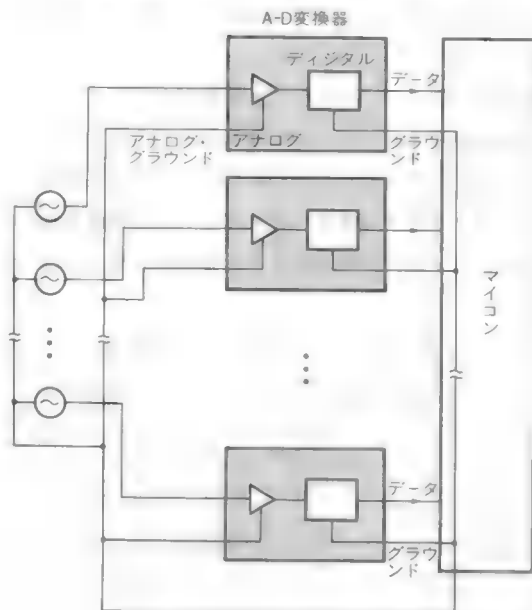


(b) バス・インターフェースとビット



(c) バスに直接接続する

図 3.40 アナログ・グラウンドの接続



ん、とくに接地線（デジタル・グラウンド）はあまり“きれい”であるとは期待できません。入力しようとするアナログ信号が 1 本だけの場合はあまりたいした問題は起きないのですが、図 3.40 のように複数の信号を接続する場合、接続方法が問題となります。

デジタル・グラウンドに 100mV くらいの電位差がつくことはザラですが、12 ビットのアナログ分解能は $10\mu\text{V}$ といったオーダーになることを忘れないでください。

(2) 基準電圧源の温度特性はどうか

図 3.39 の例では ICL7109 内部の基準電圧源を使用していますが、この温度係数は 80 ppm/°C であり、0～50°C で使うとすれば 0.4% もの誤差になります。これに対して外付けの基準電圧源を使えば 1 ppm/°C くらいまでは得られます。これならば 50°C の温度範囲に対して 12 ビットの確度を保てます。

このように、広い温度範囲で 12 ビット以上の確度を得るのは案外大変で、基準電圧源以外にも CR の温度特性などを抑え込まねばならない場合もあります（2 重積分型では CR の温度特性は原則として誤差にはなりません）。

コラムF デコーダのいろいろ

デコーダは入力が特定の条件になったことを検出すれば良い訳ですから、本質的には単なるAND (またはNAND) ゲートです。問題は、入力線数が非常に多いことです。たとえばZ80の場合、 \overline{RD} 、 \overline{MREQ} 、アドレス(の一部)を使用する必要があります。

たとえば図F.1の基本的なバス・インターフェースをアドレスF000H~F0FFHで使用するためのデコーダを考えてみます。図(a)はゲートを組み合わせたもの、(b)、(c)はアドレスをジャンパやDIPスイッチで設定できるものです。(c)のオープン・コレクタ型Ex-NORゲート(74LS266)はこのような一致検出に便利ですが、ワイヤードANDが可能なゲート数には制限があります。(d)はコンパレータを使用したもの、(e)はデコーダ用MSIを使ったもの、(f)はPAL (Programmable Array Logic) を使ったものです。コンパレータはTTL、CMOSともに種類が増えてきており、今後はもっと多く使われるでしょう。また、PALやPROMは、あらかじめ書き込んでおく必要がありますが、複雑なデコード回路でも1個のICで実現できます。

コラムG フル・デコードとリニア・デコード

メモリやI/Oポートにアドレスを割り当てるとき、必要最小限のアドレス空間しか与えないようにアドレスをデコードすることを“フル・デコード”といいます。

たとえば、4KバイトのROMを64Kバイト(2^{16})のアドレス空間に置く場合、フル・デコードするには少なくとも上位4本のアドレスをデコードする必要があります(図G.1:93ページ参照)。また、本文中の図3.6のような1バイトの入力ポートを、同じアドレス空間にフル・デコードで割り当てするには、16本のアドレスのすべてを用いてデコードを行う必要があります。

フル・デコードを行った場合の利点は、ムダなアドレスを生じないことです。アドレス空間に余裕が少ない場合や、今後の装置拡張の見通しがつかない場合、アドレス空間をなるべく有効に使用するためフル・デコードとします。

一方、アドレス空間に余裕があればフル・デコードする必要はありません(図G.2)。本文の図3.6の例では1バイトの入力ポートに256バイトの空間が割り当てられてい

図F.1 デコーダのいろいろ

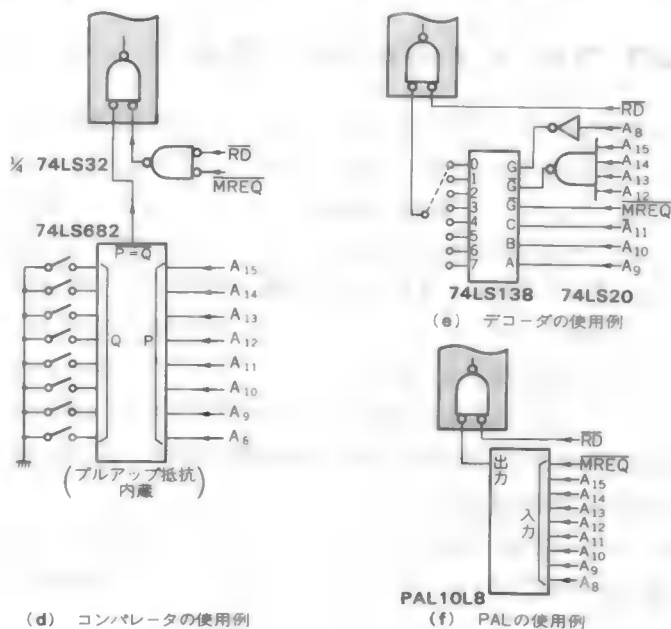
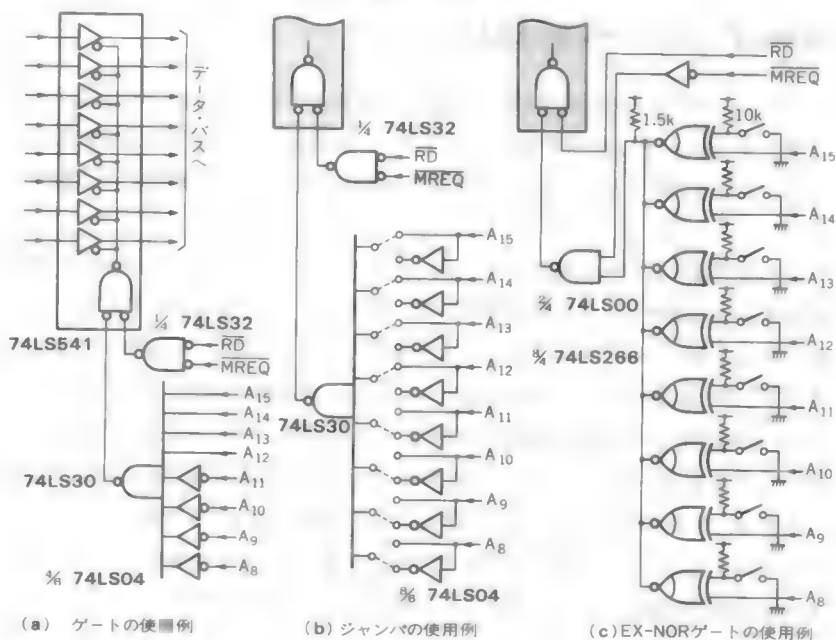


図 G.1 フル・デコードされたシステム

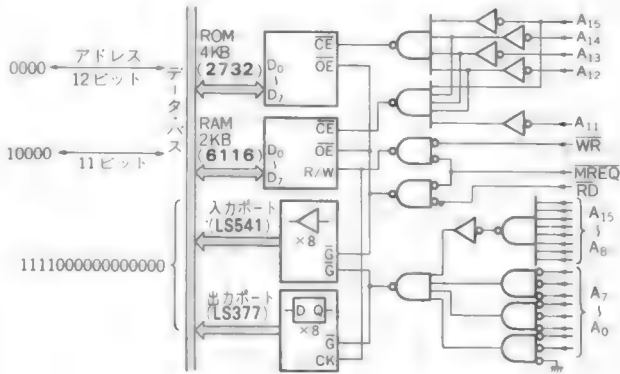


図 G.2 フル・デコードではないシステム

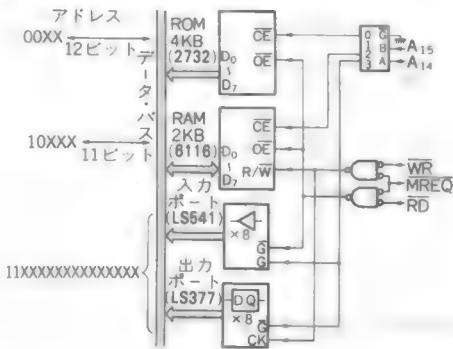
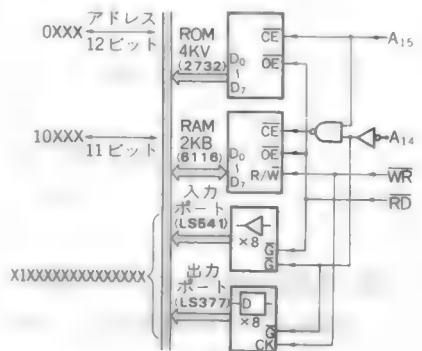


図 G.3 リニア・デコードを用いたシステム



ます。このほうがデコード回路が簡単になるからです。

この考え方のもっとも極端なものが“リニア・デコード”です。これはデコード回路をとくに設けず、アドレス線を各素子のイネーブル端子に直接に接続してしまうというもので、ハードウェアはもっとも簡単になります。ただし、ソフトウェアのミスにより出力が衝突する危険もあるので注意が必要です(図 G.3)。

コラムH スイッチの種類について

スイッチに関していろいろな分類の方法があります。ここでは、つぎの四つの分類法について示します。

図 H.1 つき合わせ接点と
すり合わせ接点

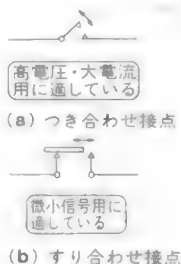


図 H.2 SPST, DPDT, SPDT など

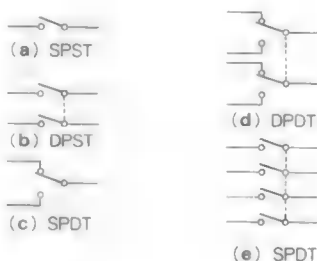
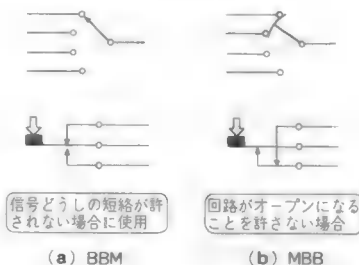


図 H.3 BBM と MBB



■つき合わせ接点とすり合わせ接点

水銀接点のような特殊なものを除き、機械的接点にはつき合わせ型とすり合わせ型とがあります（図 H.1）。

つき合わせ型はマイクロ・スイッチやトグル・スイッチに多く用いられ、大電流・高電圧に耐えることを特徴とします。欠点は微小信号に対して接触不良を起こしやすいことです。

一方、すり合わせ型はスライド・スイッチやロータリ・スイッチに多く見られます。接点が接触したままある程度の距離を徐々に移動しますので、大電流や高電圧は使用できません。しかし、接点の自己清浄作用があるため、微小信号でも接触不良を生じにくいという長所があります。

● SPST, DPDT など

SPST は Single-Pole Single-Throw の略で「単極単投」と訳されます。DPDT は Double-Pole Double-Throw の略で「双極双投」と訳します。これらの意味は図 H.

2に示すとおりです。

● **BBM, MBB**

複数の接点を切り換える場合、隣の接点に接続する (make) 前に、もとの接点との接続が切れる (break) ことが Break before Make で BBM と略されます。この逆が Make before Break (MBB) です。また、BBM は Non-Shorting, MBB は Shorting ともいいます (図 H.3)。

● **NO と NC**

スイッチになにも力を加えない状態で開いているものを Normally-Open (略して NO)、逆に閉じているものを Normally-Closed (NC) といいます。とくに NC は、Non-Conection とまちがえやすいので注意。

第4章

マイコン出力インターフェースの設計

この章ではマイコンから外部機器にデータを出力する方法を説明します。

出力用インターフェースの一般形を図4.1に示します。ディジタル信号を出力するには、普通、データ・バス上の値をDフリップフロップなどで固定します。そのあと、必要に応じてデコードなどを行い、所定の形態およびレベルに変換します。

一方、アナログ信号は一般にバス上の値を固定したあとディジタル-アナログ変換器(D-A変換器)に印加されてアナログ出力となります。

ただし、マイコンからの出力はデータ・バス上の値を固定して使うだけとはかぎりません。たとえば、データ・ストロブ信号を利用してパルス列を出力したり、アドレス信号を固定して並列データとして用いるなど、様々なバリエーションがあります。

また、発光ダイオード(LED)や液晶などの表示器の駆動回路も出力インターフェースに含めて考えるべきでしょう。

図4.1 出力用インターフェースの一般形

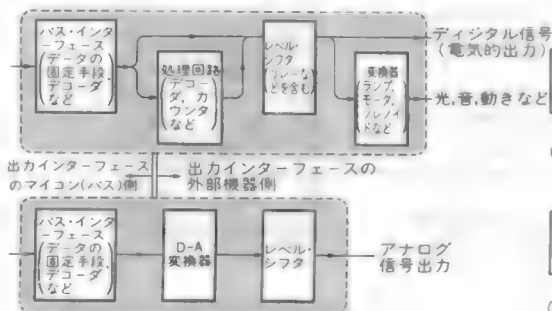
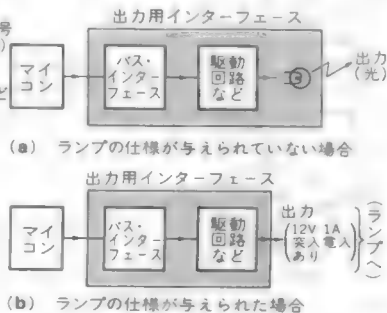


図4.2 出力用インターフェースの範囲



本章では以上のようなことから、様々な出力インターフェースの事例を紹介します。

4.1 出力信号の種類と処理方法

マイコンから出力しようとする信号にも、入力信号に匹敵する種類があります。ここでは第1章に示した各項目の内容を簡単に説明しておきます。

●出力信号の形態とレベル

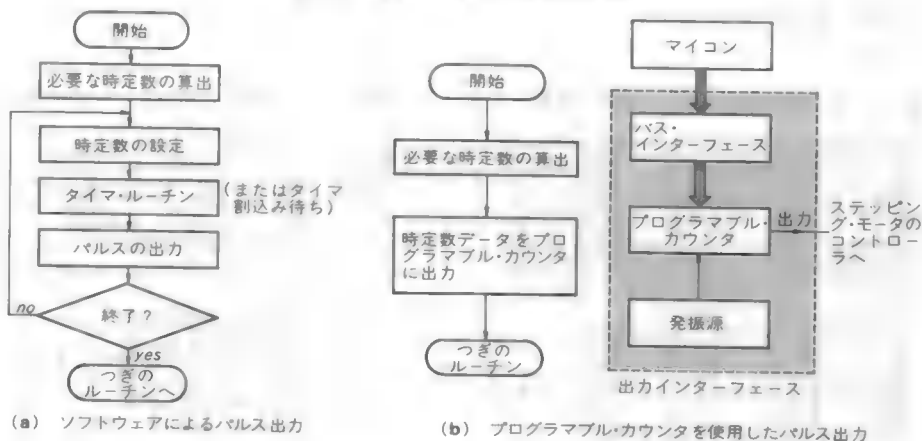
出力信号の形態やレベルの分類を行う場合、考えなければならないのはつぎの2点です。

- (1) 出力用インターフェースの範囲の考え方が一通りでない。
- (2) 出力信号は、入力信号と異なり電力を考える必要がある。

まず(1)の問題については図4.2を用いて説明します。これは、ランプを点灯する例ですが、出力信号を“可視光線を利用した表示”と考えた場合、必要な明るささえ得られれば何V何Aの電球を使おうが、あるいはLEDを使おうが、インターフェース設計者の自由です。ところが、使用する電球の定格がすでに決定されている場合、インターフェース回路の扱う信号は「電流・電圧」で示されます。

本章ではモータやソレノイドなど機械的運動への変換器を割愛する点を除き、様々な出力信号形態を取り扱います。

図4.3 可変周波数のパルス出力の例



ただし、ここで注意しておきたい点は、出力インターフェースを設計する際には、たんに与えられた仕様にこだわらず、“最終的に必要な出力形態は何か？”をつねに考えるべきだという点です。

つぎに(2)の問題を考えます。外部からの信号をマイコンが取り込む際には、受け側の入力インピーダンスがある程度高ければ、電力的な条件を考える必要はありません。しかし、出力インターフェースは外部機器に対して電力を供給したり、電力を制御したりする必要があります。したがって、信号のレベルもつねに電力を含めて(たとえば電圧と電流とを)考えなければなりません。

●出力信号の速度

出力信号の速度として、まず第1に出力信号の状態を変化させる頻度や周期を考えることにします。

たとえば、ステッピング・モータ(パルス・モータ)のコントローラに対して印加するパルス信号をソフトウェアで作れるかどうかを考えてみます。図4.3(a)に示すのは1例で、出力パルス信号の間隔はループを1回まわる時間で制限されます。この場合、大体10 kHz(周期100 μ s)程度が限界となります。これは入力信号を取り込める最短周期と同じオーダーです。

では、これ以上のパルス周波数が必要な場合、どうすればよいのでしょうか。図4.3(b)にその1例を示しますが、これはプログラマブル・カウンタを使用してソフトウェアに無関係にパルス列を出力する回路です。こうすることによって、ソフトウェアの速度とは関係なく、高い周波数のパルスでも出力することができます。

ところが、この周波数を変化させようとすれば、やはりループをまわる時間が問題になります。したがって、ソフトウェアで判断や演算を行うかぎり、一定の時間より速く出力状態を変化させることはできません。これが、出力信号の状態を変化させる周期の限界であるといえます。

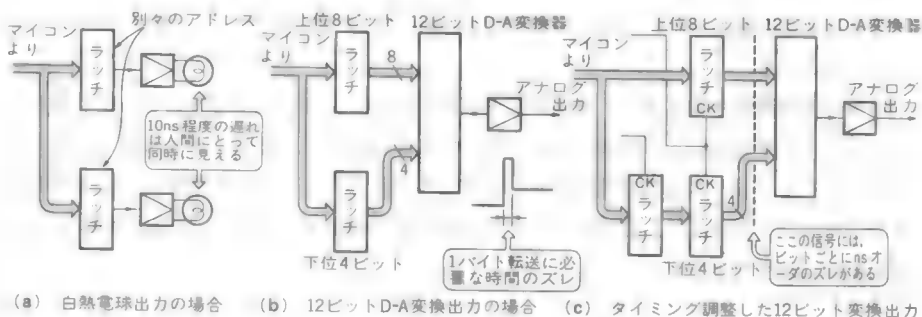
●出力信号の同時性

つぎに、出力信号の同時性について考えてみます。

複数の信号を“同時に”変化させなければならないことはよくあります。しかし、厳密な意味で“同時”というのは現実的ではありません。

たとえば図4.4(a)の白熱電球による表示の場合、人間の感覚の遅れや電球の時定数を考

図 4.4 出力信号の同時性が要求される場合



えると、10ms くらいの差で点灯してもとくに問題はありません。したがって、2個の電球に別々のアドレスが割り当てられていても、マイコンから“同時に”点灯することができます。

ところが、図 4.4(b)の 12 ビット D-A 変換器の場合、上位 8 ビットと下位 4 ビットとに別々のアドレスを割り当てておくと、アナログ出力を変更しようとするたびに大きなグリッチ（出力に生じるヒゲ）を生じ、使用目的によっては使いものになりません。

普通、このような場合には同図(c)に示すようにラッチを2段にするなどしてタイミングの不均一をなくします（例 4-4 を参照）。ところが、1個の IC の中でも個々のビットの信号が同時に変化するという保証はありません。また、D-A 変換器内部の速度のズレや配線などの影響により、各ビット間に最大で数 10ns のタイミングのズレがあり得ます。この程度の時間が問題となる場合、アナログ信号をホールドするなど、さらに別の対策が必要です。このように“同時”といってもその程度によって様々な対策が必要です。

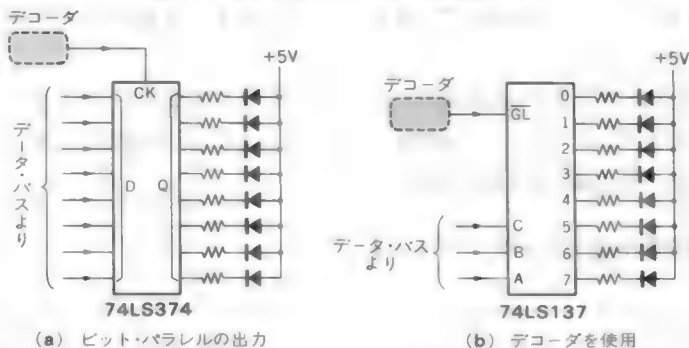
●出力信号の数

マイコンからの出力では、インターフェース回路からデータ・バスにデータを乗せることは (DMA を除けば) 普通はありません。したがってバッファさえ設ければマイコンのバスに与える影響は問題ありません（例 4-3 を参照）。

しかし、きわめて多数の信号（たとえばマトリクス状の LED の駆動信号）を出力する場合、信号線の本数や消費電力の削減のため、つぎのような事項が使えないかどうかを考えてみます。

(1) デコーダや簡単な論理演算により、マイコンのバスに接続する信号線を減らすこと

図 4.5 多数の LED のうち 1 個を点灯する



ができないか？

たとえば、多数の LED のうち 1 個だけを点灯するのであれば、図 4.5 (a) よりも同図 (b) の構成のほうが通しています。

- (2) 表示器のようなものの場合、ダイナミック駆動を行うことで外部機器側の信号線を減らすことができないか？

とくに表示関係では各種のデコーダやダイナミック駆動用 IC が入手できるので、なるべくそれらを利用するよう心がけます (例 4-9, 例 4-10 を参照)。

4.2 出力インターフェースのマイコン側の設計

外部に何らかの信号を出力するためのインターフェース回路の、マイコン (バス) 側について考えてみます。この部分はつぎのような方法が考えられます。

- (1) データ・バスの値を固定する もっとも一般的なのはこの方法です。マイコンのデータ・バスは時分割で様々な情報が乗っているので、必要とする時点の値を D フリップフロップなどで固定します。このために、アドレスや制御信号の値をデコードする回路が必要です。
- (2) 制御信号を使う パルス状の出力信号が必要な場合、アドレスや制御信号をデコードしたものを、そのまま出力として使うことができます。

この場合、データ・バスの内容は何でもかまわないので、書込み (\overline{WR}) 信号にかぎらず読出し (\overline{RD}) も利用可能です。

- (3) アドレス・バスの値を固定する CPU のアドレス・バスは出力専用のバスと見るこ

とができます。多くのCPUではレジスタ間接アドレッシングを始めとして、アドレス・バスに任意の値を出力する手段を備えていますから、データを出力する際には利用できます。

以上のように、マイコンから情報を出力するにはいろいろな方法があります。いずれの場合にも、出力インターフェース回路はマイコンのバスにとって負荷となりますから、多数の信号を出力する際には注意が必要です。

例 4-1 基本的な出力インターフェース

出力用インターフェースの基本は、データ・バスの内容を何らかの方法で固定することです。

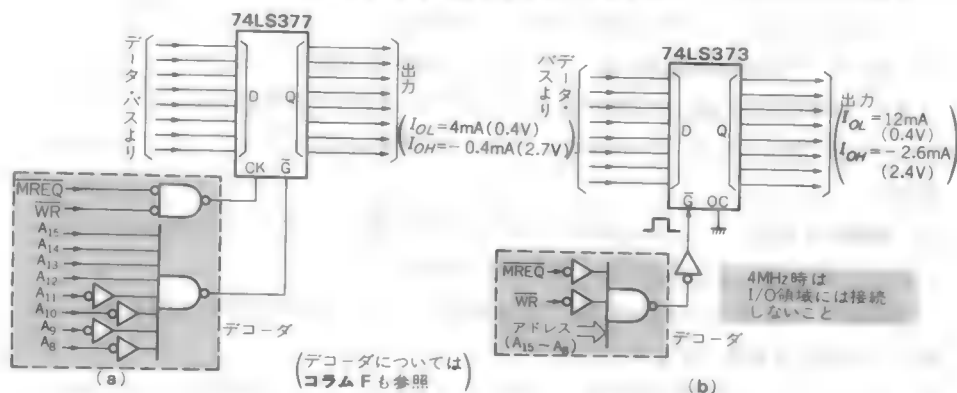
ここでは、データ・バスと内容を固定する基本的な方法と、タイミングの計算方法とを説明します。

図 4.6 に示すのは、基本的な出力インターフェースの例です。

▶デコードとデータ・バスの内容の固定 一般に出力インターフェースのマイコン側にはつぎのような要素が必要です。

- (1) アドレスや制御信号が所定の組合せになったことを検出するデコーダ。

図 4.6 基本的出力インターフェース



(2) (パルス状の出力信号でよい場合を除いて) データ・バスの内容を固定する手段。

まず、デコードについては例 3-1 を参照してください。普通は読出し信号 (\overline{RD}) のかわりに書込み信号 (\overline{WR}) を使います (あたりまえ)。しかし、出力用インターフェースはデータ・バスに信号を乗せるものではありませんから、特殊な使い方として \overline{RD} を用いることもできます。

前記(2)のデータ・バスの内容を固定する手段ですが、普通は D フリップフロップ (コラム 1 参照) またはトランスパレント (透過型) ・ラッチ (コラム 1 参照) を用います。ただし、どのタイミングでデータを固定しなければならないかは CPU によって、あるいは使い方によって異なりますので別途に検討しなければなりません (後述)。

▶ CPU に接続するための条件 つぎに、図 4.6 (b) の回路を実際に CPU に接続できるかどうかを調べてみます。この場合に考慮しなければならないのはつぎの 2 点です。

- (1) この回路を接続することによって CPU や他のバス・スレーブに与える影響。
- (2) この回路がマイコンのバス上の信号を固定できるか否か。

▶ バス・スレーブに与える影響 まず前者ですが、TTL を使用するかぎり、出力用インターフェースのほうが入力用よりもデータ・バスに大きな影響を与えます。これは、おもに TTL の入力電流に起因します。

たとえば図 4.6 (a) の 74LS377 の場合、入力電流は、

“H” 側: $I_{IH} = 20\mu\text{A} (\text{max})$ (流込み)

“L” 側: $I_{IL} = -400\mu\text{A} (\text{max})$ (流出し)

です。一方、Z80A CPU のデータ・バスの駆動能力は、

“H” 側: $I_{OH} = -250\mu\text{A}$ ($V_{IH} = 2.4\text{V}$)

“L” 側: $I_{OL} = 1.8\text{mA}$ ($V_{OL} = 0.4\text{V}$)

ですから、他のバス負荷がなにもなくても、この回路は 4 個しか接続できません。

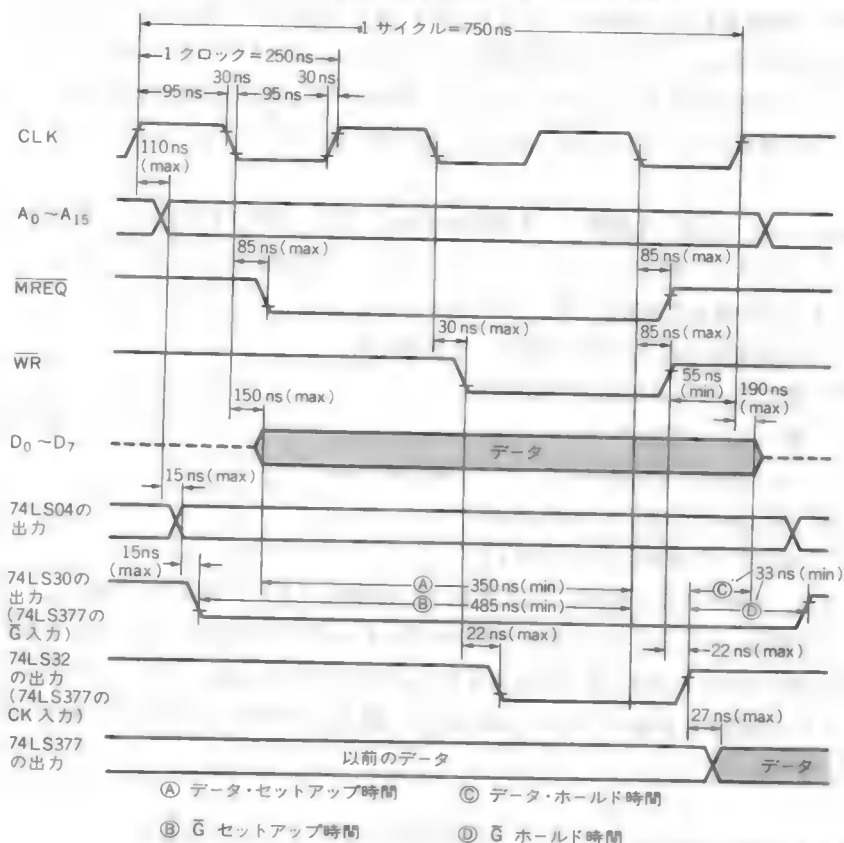
メーカー製のパーソナル・コンピュータではデータ・バスにバッファが入っているので、CPU に対してはこれほどシビアではありませんが、うっかりと他のバス・スレーブ (メモリなど) の駆動能力を越えてしまうことがある (図 2.6 を参照) ので充分に注意しなければいけません。

▶ バス上の信号の固定法 つぎに、この回路がマイコン・バス上の信号を正しく固定で

きるか否かを調べてみます。

図4.7に示すのはZ80A CPUのメモリ領域に図4.6(a)の回路を接続した④の具体的なタイミングです。Z80のメモリ領域ではデータが出力されてから \overline{WR} の前縁(立下り)まで1クロック、後縁(立上り)までならば2クロックの時間があります。各バスにバッファが挿入されているか否か、あるいはデコード回路の形によって多少の変化はありますが、Z80A CPU(4 MHz)のメモリ領域で \overline{WR} 信号の後縁を使えば、大体300ns以上のデータ・セットアップ時間が確保できます(図4.7の場合は④の350ns)。また、データ・ホールド時間もかなり得られます(③33ns)。これはTTLのDフリップフロップでデータを固定す

図4.7 Z80Aを用いたときのデータ出力タイミング



るには十分な値ですし、トランスバレント・ラッチも使用可能です。

一方、同じ Z80A でも I/O 領域では \overline{WR} の前縁に対してはデータ・セットアップ時間が確保できません。したがって、図 4.6(b)の回路は I/O 領域には使用できません。*

例 4-2 ビット単位でオン/オフできる出力インターフェース

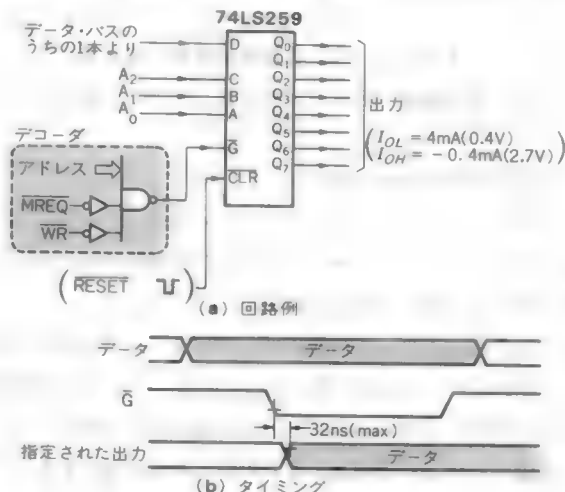
他の出力信号の状態を変化させることなく、1本の出力信号だけを変化させたいことがあります。

このような場合、データ・バス中の1ビットだけを使用して各信号を別々のアドレスに振り分けるのが簡単です。

リレーやランプを駆動する場合、1個ずつ独立にオン/オフできたほうがよいこともあります。そのような場合、図 4.8 に示すようにアドレスブル (ビット指定可能)・ラッチを使用する方法があります。

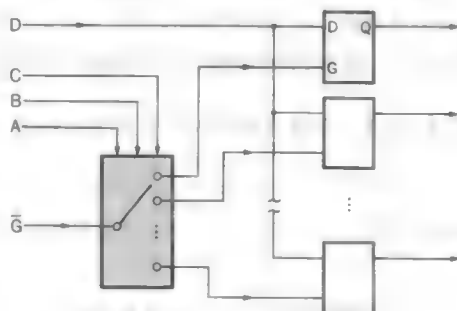
アドレスブル・ラッチは図 4.9 のようにデコードと各ビットのラッチとが組み合わせられ

図 4.8 ビット単位でオン/オフできる出力インターフェース



* クロック ■ 数やそのデューティ比を選ばば、そのかぎりではない。

図 4.9 アドレスラッチの内部構成



たものです。図 4.8(a) の 74LS259 は TTL では代表的なアドレスラッチで、 \bar{G} が“L”の時点では D に印加したデータが出力 ($Q_0 \sim Q_7$) のうちの 1 本 (3 ビットのアドレスで指定する) につつまけになります。つまり、ラッチ機能としてはトランスパレント・ラッチですので、出力不確定の時間が生じないように注意します。たとえば、4 MHz の Z80A CPU の I/O 領域には、このまま直結すると出力不確定の時間が生じ得ます。

例 4-3 多数の信号を出力するインターフェース

出力用インターフェースはバスに与える影響が無視できません。したがって、多くの信号を出力しようとする場合、データ・バスにバッファを設けるなどの方法が必要となります。

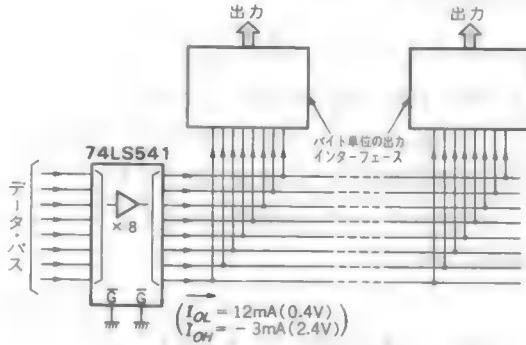
ここでは、そのような例を説明します。

基本的な出力インターフェースはデータ・バスに信号を送り出す必要がないので、単純にバッファを挿入すれば済みます (図 4.10(a))。

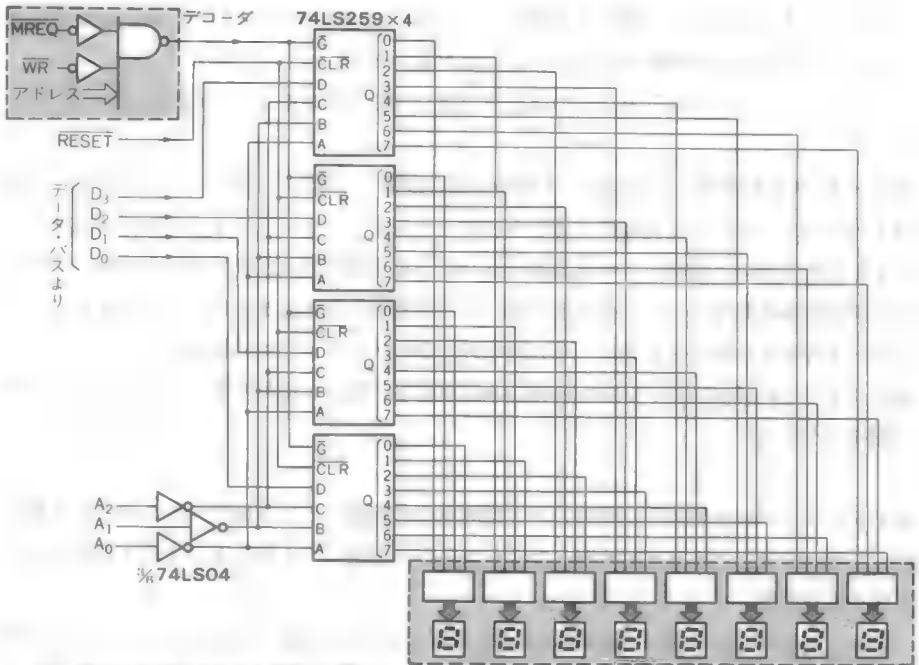
また、前述のアドレスラッチを使えば、データ・バスの負荷を軽くできます (図 4.10(b))。ただし、アドレス・バスの下位 3 ビットには、バッファが必要となります。

次項に示すように CMOS IC を用いれば、バスの直流負荷を減らすことができます。しかし、交流負荷 (容量分) はむしろ増加しますので、考慮を要します。

図 4.10 多数の信号を出力する



(a) バッファを挿入しただけの例



(b) アドレスラッチを使用した例

7セグメント・デコーダおよびLED駆動回路(8桁分)

例 4-4 多数の信号の同時出力

多数の出力信号を同時に変化させなければならない場合があります。

ここでは、データ・バス幅よりも多くの信号を同時に出力する方法として、

(1) データ・バスを複数回使用してラッチを2段とする

(2) データ・バスとアドレス・バスとを同時に使用する

の2通りについて説明します。

データ・バスのビット数よりも多くの信号（たとえば8ビット CPU に対して9本以上の信号）を同時に変化させなければならない場合もあります。

代表的なのは D-A 変換器で、すべてのビットを同時に変化させないと、データ変更のたびにグリッチを生じます（図 4.4 を参照）。このほか、リレーやアナログ・スイッチの制御の際にも信号変化の同時性が要求されることがあります。

ただし、ここで“同時”というのは信号のズレの許容量が数 10ns の場合を指します。

▶データ・バスを使用した多数ビット同時出力の回路 図 4.11 (a)～(c)に示すのは、2語以上のデータをまとめて同時に出力するバス・インターフェースです。同図(a)は最上位バイトを書き込むと同時に、それ以前の全バイトを2段目のラッチに固定するものです。データ書き込みの順序として、必ず最上位バイトを最後に書き込まなければなりません。

データの書き込み順序を任意にしたいのなら、2段目のラッチへの転送ストローブを別に設けます。これは他のアドレスへの書き込み信号でも、読出し信号でも、なんでも使えます（図 4.11 (b), (c)）。

▶アドレス・バスを使用した多数ビット同時出力の回路 以上はデータ・バスを複数回使ってデータを出力する例ですが、アドレス・バスを用いて1語以上のデータを出力する方法もあります。

表 4.1 に示すのは、Z80、6809、8088 の各アドレス・バスを使って出力できるデータの例です。いずれの CPU もレジスタを使った演算の結果をアドレス・バスに出力することができますから、見方によっては24ビット（Z80、6809）または28ビット（8088）までの信号は同時に出力可能です。

図 4.11 多数ビットの同時出力

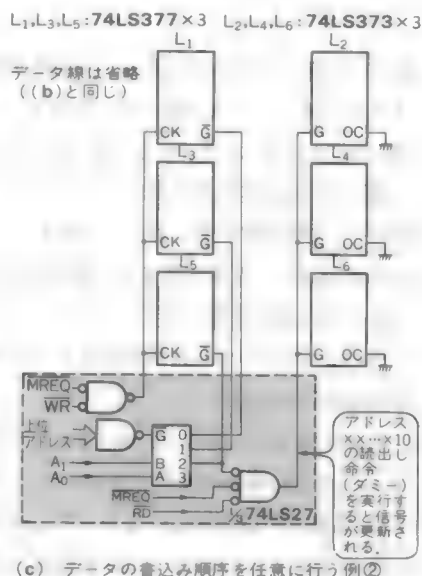
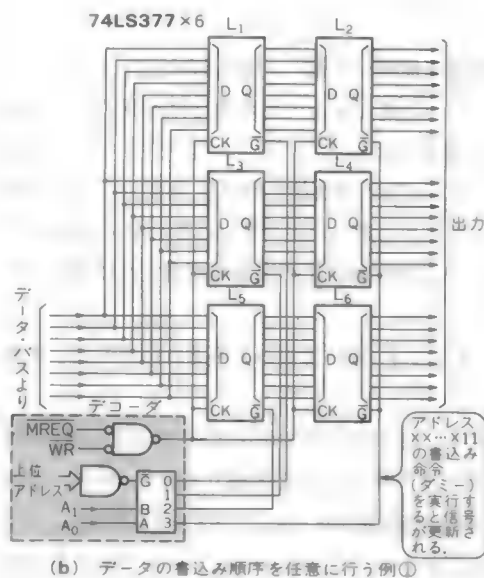
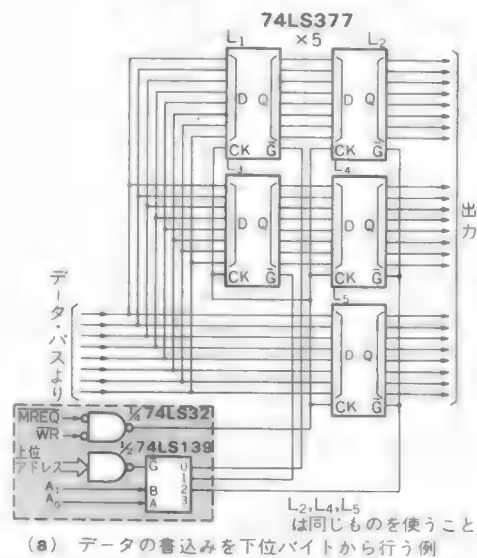


表 4.1 アドレス・バスの内容を出力として使用する

CPU	命令(例)	アドレス・バス	データ・バス
Z80	LD (nn), A	M イミディエイト	A
	LD (HL), A	M HL	A
	LD (BC), A	M BC	A
	OUT(BC), A	IO BC	A
6809	STA (ダイレクト)	DP + イミディエイト	A
	STA (B オフセット)	X + B	A
8088	MOV メモリ, A	DS	AL
		BX + DI	

図 4.12(a)は Z80 のアドレス・バスを使って 12 ビットのデータを一度に出力するものです。この場合、少なくとも 4 K バイト分のアドレスを占有しますから、アドレス空間に余裕がないと使えません。

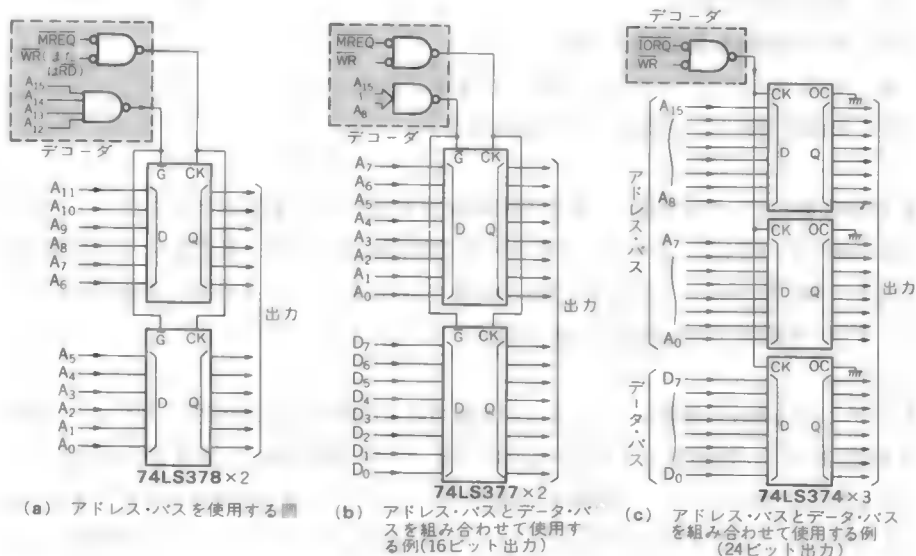
▶アドレス・バスとデータ・バスを組み合わせて使用した例 図 4.12(b)はアドレス・バスとデータ・バスとの組合せによるものです。計 16 ビットの信号ならば占有アドレスは 256 バイト、計 12 ビットならば 16 バイトに過ぎません。

図 4.12(c)は Z80 のアドレス・バスとデータ・バスとの全幅を用いた 24 ビットの出力回路です。全幅を使ってしまっは困るのではないと思われるかもしれませんが、RAM や他の出力ポートをすべてメモリ領域に置き、入出力領域にこの回路を配置すれば可能です(もちろん逆の配置も可)。

このように、アドレスに余裕さえあれば、8 ビット CPU であってもそれ以上のビット数の同時出力が可能です。

なお、図 4.11、図 4.12 のいずれにおいても、D フリップフロップ(またはラッチ)の各ビットの遅れ時間の差は出力タイミングのバラツキとして必ず残ります。LS タイプの TTL の場合、標準的には数 ns、74LS377 の最大値は 27ns ($V_{CC} = 5V$, $T_a = 25^\circ C$) です。これが問題となる場合、D-A 変換器であればアナログ信号をホールドする方法があります。

図 4.12 アドレス・バスを使って多数ビットを出力する方法



例 4-5 CMOS 出力インターフェース

CMOS IC には入力インピーダンスが高く、消費電流が少ないなどの利点があり、CMOS によるインターフェース回路も有用です。

しかし、入力容量の増加やタイミング、論理レベルなど、考慮しなければならない問題が多くあります。

本項では、これらの問題について説明します。

例 4-1 に示したように、CPU から出力される各信号はタイミング的に余裕があります。したがって、TTL に比べて速度の遅い CMOS でも使用できる可能性があります。

出力インターフェースに CMOS を使った場合の利点としては、つぎのようなことが考えられます。

- (1) バスに対して直流的な負担が少ない。
- (2) 消費電流が少なくすむ。
- (3) 出力電圧が電源電圧いっぱいになる。

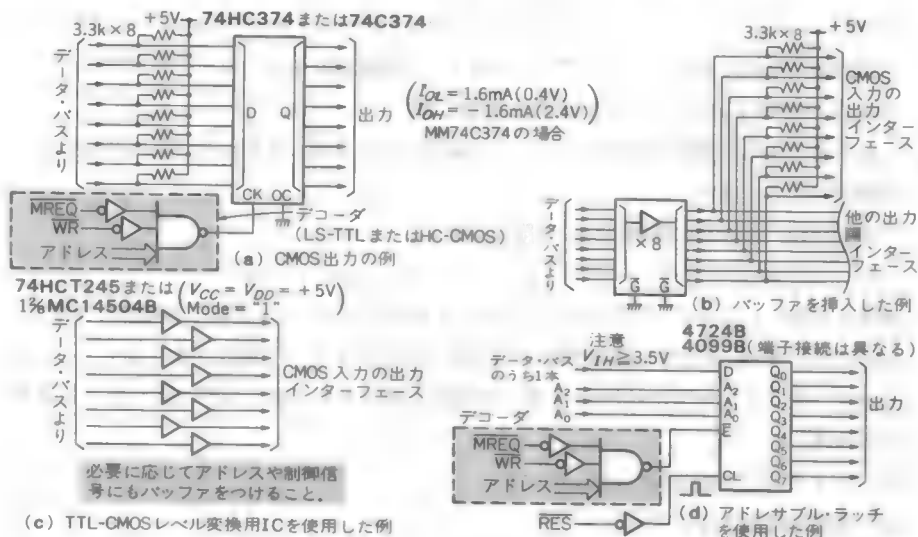
一方、CMOSがTTLに比べて不利な点はつぎのとおりです。

- (4) 速度に余裕がない。
- (5) 出力の駆動能力（電流）が小さい。
- (6) TTLレベルのバスからの“H”レベル電圧 V_{IH} が不足する。
- (7) 直流負荷が少ない反面、入力容量が大きい。

▶ CMOSを使用した出力例 図4.13(a)に示すのはCMOSを使った出力インターフェースの例です。図4.7に示すようにセットアップ時間は充分にありますから、74HC374のかわりに74C374を使っても充分に間に合います。ただし、ホールド時間の余裕は少ないので、デコード関係は手を抜かないほうが良いでしょう。

▶ “H”レベル電圧の確保 むしろ、設計時の最大の問題は V_{IH} の確保にあります。Z80や6802/09など、NMOSのCPUの出力の“H”レベルは案外低く（無負荷で3V以下のこともある）、軽いプルアップ抵抗（10k Ω 以上）をつけてもあまり電圧が上ががりません。もちろん重いプルアップを設ければ“H”レベルは上がりますが、ただでさえ不足気味の出

図4.13 CMOSを使った出力用インターフェース



力駆動能力をムダ使いすることになります。

TTL のバッファが付いている場合 (図 4.13 (b)) は、プルアップ抵抗で“H”レベルを引き上げることがむしろ容易です。なお、TTL から CMOS への“H”レベル確保のためこのようにプルアップ抵抗を付けることがよく行われ、多くの場合、とくに問題なく動作します。しかし、TTL の出力をプルアップしたからといって $V_{OH}(\min)$ が高くなることが保証されているわけではありませんから念のため。

もっとも確実な方法は、MC14504B や 74HCT シリーズのように TTL → CMOS のレベル変換用 IC を挿入してしまうことでしょう (図 4.12 (c))。

図 4.13 (d) に示すのは例 4-2 に相当するアドレスラッチです。4724B は 74LS259 と同じ端子接続としたものです。ただし、リセットの極性が逆 (アクティブ“H”) になっていますので注意なくはいけません。

例 4-6 バルス信号の出力

外部機器のなかには、ステッピング・モータのコントローラのように、パルス状の信号を必要とするものがあります。これ以外にも、ハンドシェイク用のストロブやアクノリッジ信号、フリップフロップのリセット信号、警報音を出すためのスピーカへの信号、電磁カウンタへの信号など、パルスの個数または周波数が問題となる用途は少なくありません。

このような場合、一定のレベルの信号を出力するのとは異なった方法も使えます。ここでは、それらについて説明します。

例 4-2 で述べた回路を用いてソフトウェアで“1”と“0”とを交互に出力すれば、確かにパルス信号が出力できます。セントロニクス型出力のストロブ信号や、スピーカからブザー状の音を出す場合など、簡単にはこの方法が使えます。

しかし、この方法では CPU がパルス間隔やパルス量の面倒を全部みてやる必要があり、周波数の上限も数 10 kHz とまりでしょう。

■データ・ストロブを直接使用する方法 図 4.14 (a) に示すのは CPU のデータ・ストロブを直接に用いるものです。データ・ストロブ信号としては、アドレスをデコードしただけというのは感心しません (アドレス各ビットの遅れ時間のズレがあるため)。パルス

図 4.14 パルス出力用インターフェース

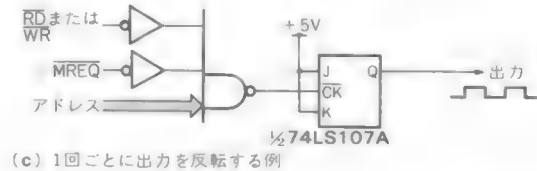
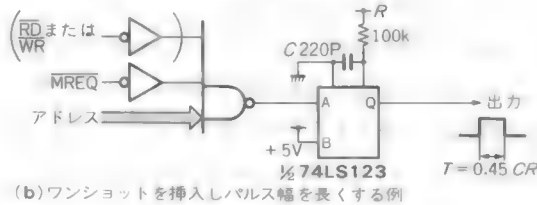
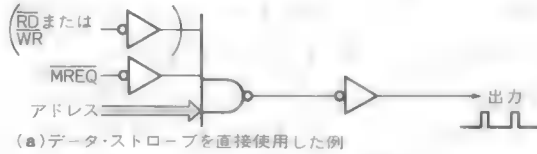
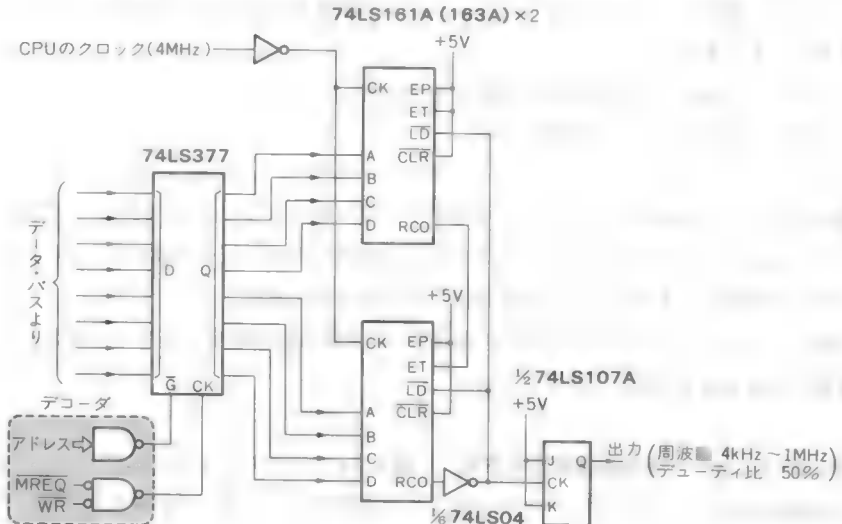


図 4.15 プログラマブル・カウンタを用いた出力インターフェース



幅は $\overline{\text{RD}}$ または $\overline{\text{WR}}$ 信号で決まり、4 MHz の Z80A のメモリ領域では 375ns (min) となります。

図 4.14 (b) はパルス幅を延長するのにワンショットを付加したもの、同図(c) は 1 回ごとに出力が反転するものでデューティ比 50 % のパルスが必要な場合 (スピーカを鳴らすには、パワ的に 50 % が有利) に使います。ただし、これらの回路は雑音に弱い欠点があることを承知しておく必要があります。

▶ **プログラマブル・カウンタを使用する方法** 以上の回路では出力パルスの周期は、やはり CPU のループ周期で決められてしまいます。それに対して図 4.15 の回路は CPU から分周比を定めるものです。8253 や Z80 CTC は基本的にはこのような回路となっています。直列通信回線のボーレート用クロック発生や、簡単なスピーカ駆動、あるいはステッピング・モータの速度制御用など、用途は広いでしょう。

例 4-7 ストロープ付き出力とセントロニクス型ハンドシェイク出力

プリンタなどのインターフェースは 2 線または 3 線式ハンドシェイクとなっています。

本項では、基本となるストロープ付き出力と、セントロニクス型ハンドシェイク出力について、実際の構成法を紹介します。

マイコンのバス自体がストロープ付き出力の形になっていますから、ストロープ・パル

図 4.16 ストロープ付き出力

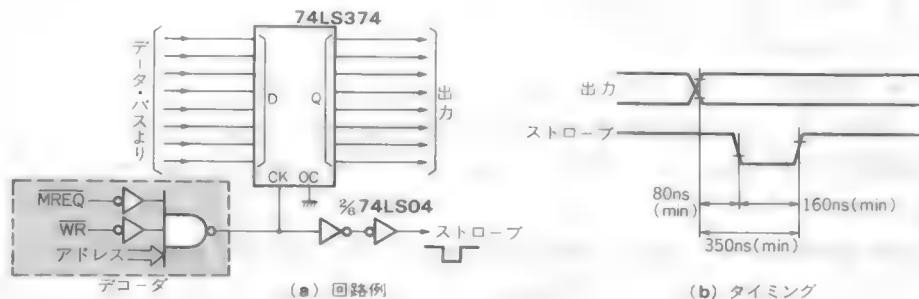
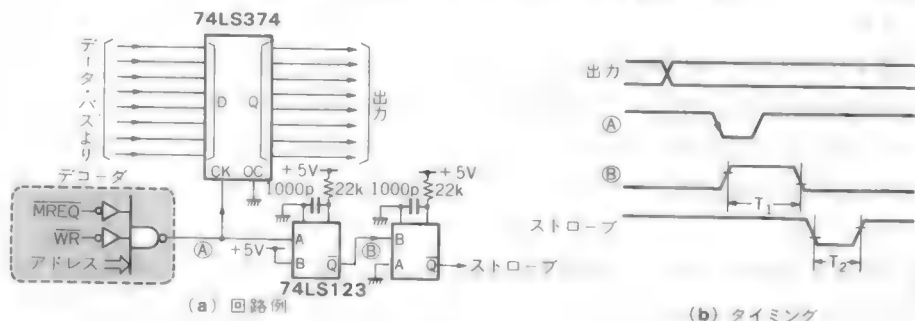


図 4.17 タイミングを延長できるストローブ付き出力



ス幅やセットアップ時間、ホールド時間が要求を満たすのであれば、図 4.16 の回路がそのまま使えます。また、図 4.17 のようにすればタイミングを延長できます。

▶セントロニクス型インターフェース 一方、いわゆるセントロニクス型インターフェース（2線または3線式ハンドシェイク）は、プリンタやプロッタの多くが採用しており、マイコンからの出力としてよく使用されます。ただし、セントロニクス型といっても実際の個々の仕様は変化に富んでおり（コラムJ参照）、すべてに適用できる出力回路は作り得ません。したがって、相手側の規格をよく調べて、それに合致するものを設計するようにします。

図 4.18 (a) はもっとも簡単なもので、市販のメーカー製パーソナル・コンピュータの多くもこの程度の回路となっているようです。この回路では、データを出力することにつきのようなプログラムが必要となります。

- (1) $\text{BUSY}/\overline{\text{READY}}$ が“L”かどうかを調べ、“H”ならば“L”になるのを待つ。
- (2) データ（8ビット）を出力する。
- (3) STROBE を“L”にする。
- (4) STROBE を“H”にする。

上記(2)、(3)、(4)は、相手側の規格によっては待ち時間を付ける必要があります。とくに $\overline{\text{READY}}$ が“L”になってから $\overline{\text{ACK}}$ が出力されるプリンタ（コラムJを参照）では注意を要します。

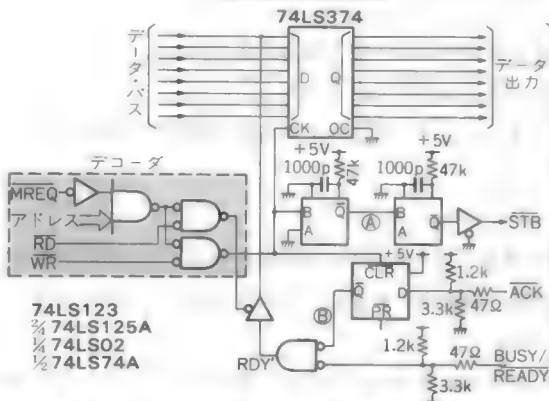
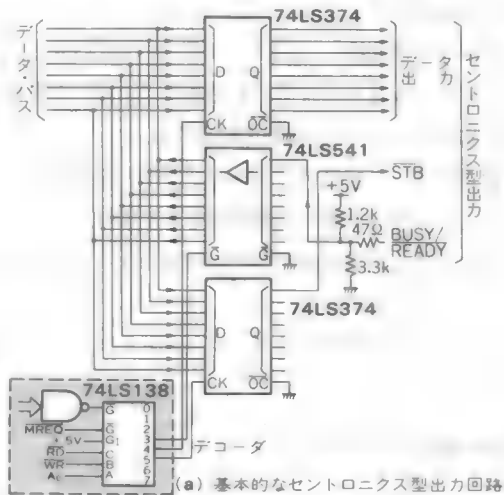
図 4.18 (b) はストローブ信号を自動的に出力するとともに、 $\overline{\text{READY}}$ を確認するためフリップフロップを設けています。この回路は、ワンショットの時定数さえ適切に選べば“セ

ントロニクス型”の大半をカバーできます。データの出力手順はつぎのとおりです。

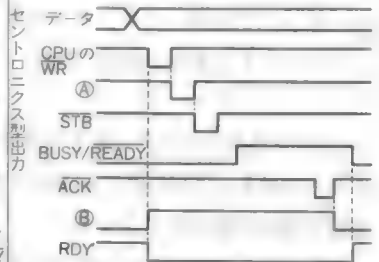
- (1) RDY 信号が“H”であるかどうかを調べ、“L”ならば“H”になるまで待つ。
- (2) データを出力する。

なお、この手の TTL レベルの並列のインターフェースを数 m も引っ張るには、受け側で確実に終端（またはプルアップ）する必要があります。プリンタなどのなかには、この処置が行われていないものがありますし、逆に市販パソコンの側の駆動能力が充分でない

図 4.18 セントロニクス型ハンドシェイク出力



(b) 自動ストローブ信号出力、READY確認つきセントロニクス型出力回路



(c) (b)のタイミング

ものもあるので注意したいものです。8255 や 6821, Z80 PIO など直接に数 m のケーブルを駆動するのは無理があります。

4.3 出力インターフェースの外部機器側の設計

出力インターフェースのマイコン側ポートによって、バスから必要な信号を取り出します。この出力は、TTL または CMOS レベルで数 mA の駆動能力をもっています。しかし、出力しようとする信号には多くの種類があり、様々な回路を付加することが必要となります。このような外部機器各ポートは、つぎのような役目をもっています。

- (1) 信号の形態およびレベルの変換 TTL レベルや CMOS レベルの信号を、接点やオープン・コレクタ、あるいは大電力のスイッチ回路など、様々な形態に変換する。また、アナログ出力が必要な場合には D-A 変換器を接続する。
- (2) 信号の多重化などを行う 発光ダイオード (LED) や液晶表示器 (LCD) には、ダイナミック駆動を用いたほうが信号線の本数が少なくすむ。また、並列データをビット・シリアル (直列) に直したほうが長距離の伝送に好都合なこともある。このような操作も出力インターフェースで行う必要がある。

本章では、以上のような点を念頭に置いて、出力インターフェースの外部機器側の例を以下に紹介することにします。

例 4-8 ランプ (白熱電球) の点灯

白熱電球は表示器として優れた特性をもつ反面、突入電流が存在するとともに、高電圧や大電流を取り扱わざるを得ないことも少なくありません。

リレーや SSR (ソリッド・ステート・リレー) を用いずに電球を駆動する場合の手法について、本項で説明します。

LED や液晶などの表示素子も急速に進歩していますが、見やすさの点では白熱電球を越えていません。そのため、短寿命・発熱・低効率・大型などの欠点にかかわらず、未だに電球も随所で使われています。

白熱電球を点灯するには電流を流しさえすれば良いのですが、電球のフィラメントは低温 (消灯) 時と高温 (点灯) 時とで、その抵抗が 1 桁以上も変化します。したがって点灯

の瞬間に定格の数十倍の突入電流が流れることになり、その対策がポイントとなります。

図 4.19 に示すのは 12V 50mA 程度の電球の点灯回路です。突入電流対策の基本は、

- (1) 消灯時にも若干の電流を流しておく
- (2) 点灯時には直列に抵抗を挿入して電流を制限する

のいずれか、または両方を行うことです。(1)の消灯時の電流はフィラメントがかすかに赤熱する程度、定格電流の 10% のオーダを限度としますが、それでも突入電流を数分の 1 から 1/10 程度にまで減らせます。

(2)については効率の悪化を無視すれば、定電流駆動としてしまえば理想的です。

図 4.20 は 12V 1A 程度の駆動回路です。電流が増加すればダーリントンにしたいところですが、飽和電圧が高くなって損失が増し、効率は低下します。パワー MOS FET(図 4.19(c), 図 4.20(b))はデジタル回路との相性が非常に良く、また突入電流やサージ電圧に

図 4.19 白熱電球の駆動——その 1

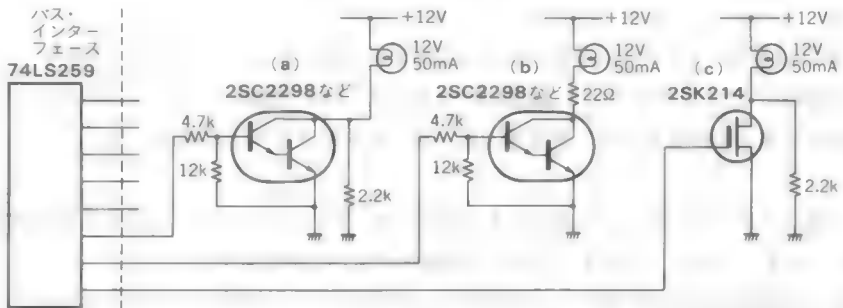
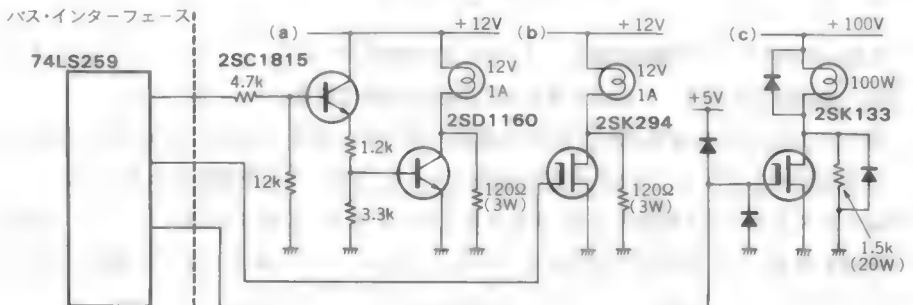


図 4.20 白熱電球の駆動——その 2



強いので、もっと使われてもいいと思います。

図4.20(c)に示すのは100V 1～2Aの駆動回路です。このクラスになるとリレーやSSR(ソリッド・ステート・リレー)を介することが多くなりますが、このように直接の駆動も不可能ではありません。ただし、マイコン側に高圧が絶対にまわり込まないよう、駆動用トランジスタの耐圧や絶縁、結線方法などに充分に注意する必要があります。

例4-9 LEDの■

LED(発光ダイオード)は高輝度化、多色化、大型化が進み、電球にかわって表示用として広く用いられるようになりました。一方、データ伝送用としても高速・高効率化が進んでいます。

もともと、LEDはTTLなどのデジタルICと接続が容易なのですが、多桁表示やデータ伝送用の高速駆動にはそれなりの注意が必要となります。

LEDは電球にかわって広く使われるようになった表示素子です。形態も単体のランプ、1次元のアレイ、7セグメントまたは16セグメントの文字表示用、5×7などのマトリクス状など多くの種類があり、色も赤・橙・黄・緑があります(青もポチポチ入手できます)。

一方、テレビのリモコンや光ファイバ通信など、データ伝送用としても重要な位置を占めています。こちらは可視光のほか、赤外領域のものもあります。

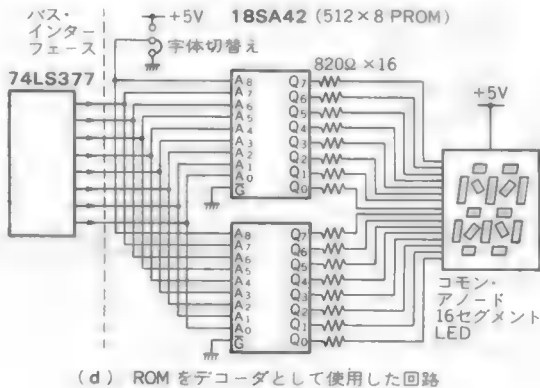
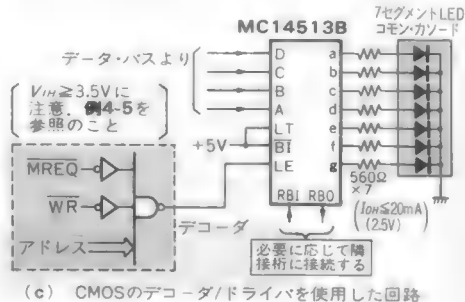
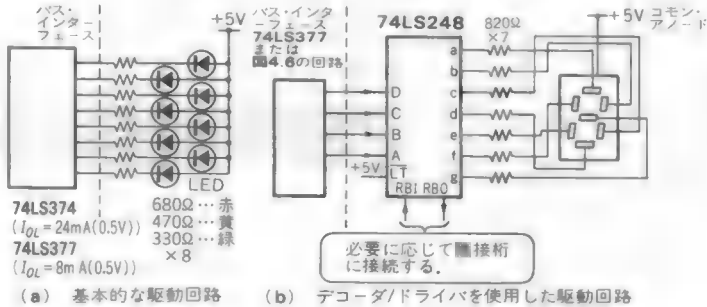
信号の絶縁用に使われるフォト・カブラの1次側も多くはLEDです。このようなデータ伝送用のLEDはオン/オフの速度や発光効率が問われることが多く、人間に見えればよいという表示用とは多少異なります。

■ LEDのスタティック■回路 ■ LEDは半導体素子であり、1～3V、1～100mAの電圧・電流があればよく、応答も速いので駆動は比較的簡単です。

図4.21(a)に示すのは基本的なLEDの駆動方法です。直径5mm以下の赤色のLEDランプを点灯するには1～20mAの電流を流せばよく、TTLで直接駆動できます。なお、このようなスタティック駆動の場合、黄色は赤に比べて1.5～2倍、緑は赤に比べて2～4倍の電流を流さないと視覚的に同じ明るさに見えません。とくに電流の小さい領域で差が大きくなります。

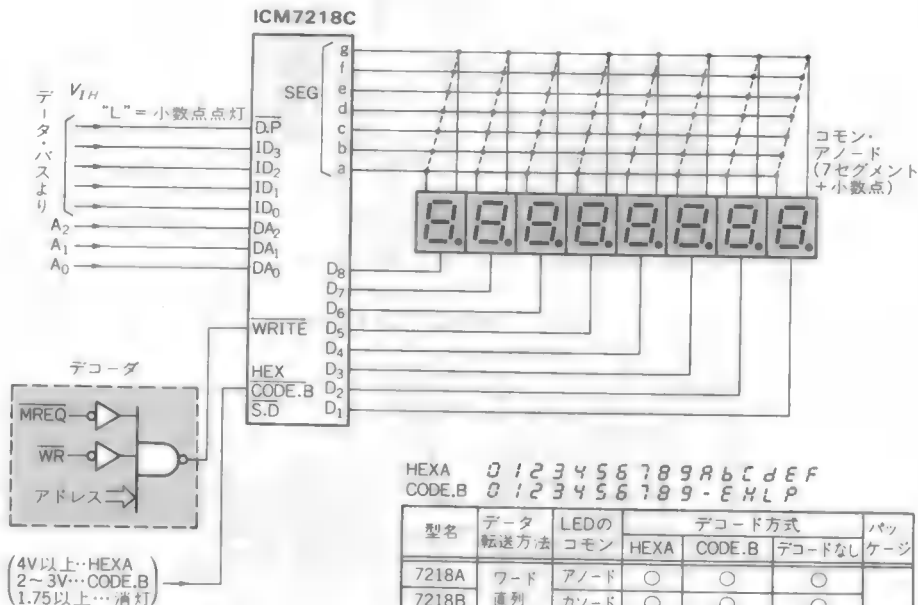
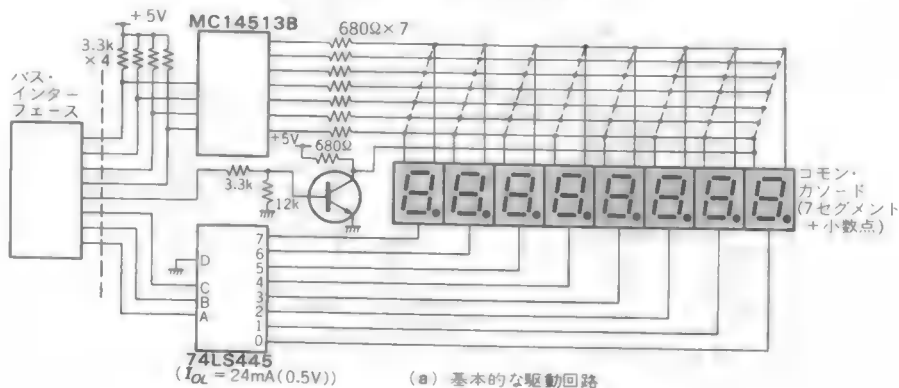
7セグメントの数字表示器には専用のデコーダ用ICがあります。図4.21 (b)はTTLのデコーダ/ドライバを使用した例です。字体と駆動能力とによって多くの種類があります。一方、同図(c)はCMOSのデコーダ/ドライバを使用した例です。ラッチ付きのものもあ

図4.21 LEDのスタティック駆動



りますので、CPU のバスに直結も可能です。CMOS で駆動能力が不足する場合は TTL やトランジスタで補強します。

図 4.22 LED のダイナミック駆動



HEXA 0 1 2 3 4 5 6 7 8 9 A B C D E F
CODE.B 0 1 2 3 4 5 6 7 8 9 - E H L P

型名	データ 転送方法	LEDの コモン	デコード方式			パッ ケージ
			HEXA	CODE.B	デコードなし	
7218A	ワード	アノード	○	○	○	28
7218B	直列	カソード	○	○	○	
7218C	アドレス 指定	アノード	○	○	—	
7218D		カソード	○	○	—	
7218E		アノード	○	○	○	40

16セグメントのものは数字および英文字が表示できます。しかし、7セグメント用のような簡便なデコーダICが入手できませんので、図4.21(d)のようにROMを使ってデコーダを作ると良いでしょう。もちろん、ソフトウェアでデコードしてもかまいません。

▶ **LEDのダイナミック回路** 図4.22に示すのはダイナミック駆動の方法です。一般に、LEDも個数が多くなるとバス・インターフェースも配線も大変です。したがって、人間の眼の残光特性を利用したダイナミック駆動によって駆動信号の本数を減らします。

図4.22(a)に示すのは基本的なダイナミック駆動の回路です。これはソフトウェアによって一定時間ごとにCPUから値を書き込む必要があります。この周期は20ms以内にしないと、点滅をはっきり感じます。また、スキャンが速すぎると回路やLEDの遅れによって接桁が点灯します。本当は表示桁を切り換える際にブランキング期間を置くべきです(図4.23)。

図4.22(b)はダイナミック駆動の専用ICを用いた例です。このICでは64個までの単体のLEDランプ、または(7セグメント+小数点)×8桁の数字表示器を駆動できます。データの転送方法、LEDのコモンがアノードかカソードか、デコードするしないかにより、7218A~Eまでの5種があります。CPUからはたんなる出力ポートに見えますが、内部メモリの読出しはできません。

▶ **データ伝送用LEDの駆動回路** つぎに、データ伝送用のLEDの例を示します。図4.24(a)はフォト・カブラの1次側の駆動例です。とくに高速用でないフォト・カブラの場

図4.23 ダイナミック駆動時のブランキング期間

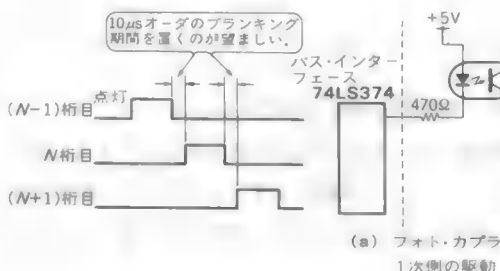
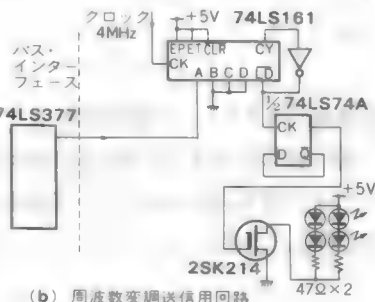


図4.24 データ伝送用LEDの駆動



合、内蔵されているのは近赤外線 LED で、順方向電圧は 1.5~2 V です。

同図(b)に示すのはデジタル信号により周波数変調 (F1) を施して近赤外線を空間伝搬させるものです。この種の LED はテレビのリモコンや煙報知機の光源として量産されており、発光率も高く、50mA の電流を流すと 10mW 程度の光出力が得られます。しかし、立上り・立下りの速度は 0.5~1 μ s と遅いので、変調周波数は 300kHz ほどでしょう。なお、図 4.24 (b)の送信部に対して、pin フォト・ダイオードを用いた FM 受信部を組み合わせると、1200bps 程度の信号を 10~20m 空間伝送できます。

例 4-10 液晶表示器の■

液晶表示器は低消費電力で見やすく、多くの場所で使われようとしています。しかし、交流で駆動する必要がある、応答が遅くてダイナミック駆動がむずかしいなど、特有の問題がいくつかあります。

ここでは、液晶表示器の基本的な駆動方法を示します。

液晶表示器* (LCD) は消費電力がきわめて小さく、低い電圧で作動し、明るい場所でも見やすいなど、多くの利点を持ち、時計、電卓、ハンドヘルド型パーソナル・コンピュータなど、電池動作の機械にはなくてはならない存在です。

一方、液晶自身是一种の電解質であり、温度変化や紫外線照射に弱いことのほかに、直流の印加ができない(電気分解して気泡を生じる)ことや応答速度が遅いためダイナミック駆動がむずかしいなどの駆動回路上の問題点があります。

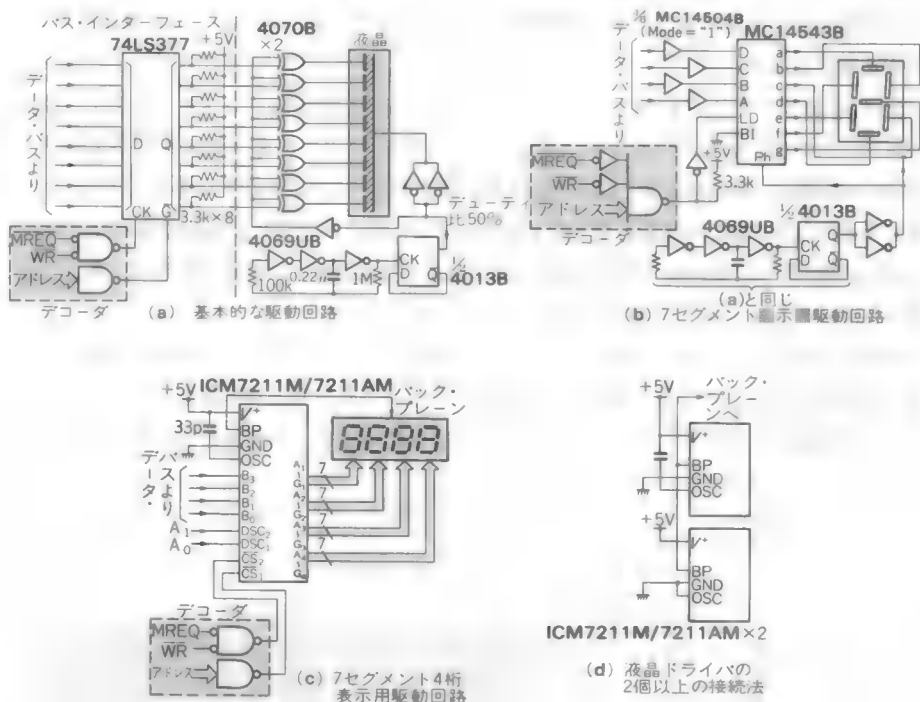
また、液晶の用途は低消費電力が要求される場合が多いため、駆動回路も消費電力を小さくするよう工夫します。なお、液晶自体は発光しませんので、暗い場所では照明が必要です。

■液晶のスタティック駆動回路 つぎに、実際の液晶駆動回路を示します。

図 4.25 (a)に示すのは基本的な液晶駆動回路です。液晶に直流分が加わらないよう、デューティ比 50 % の方形波を用いて両端をプッシュプルで駆動します。この周波数は液晶表示器にもよりますが、20~60Hz のものが普通です。点灯時に液晶に印加される電圧は、電源

* ここでは電界で作動するものをいう。

図 4.25 液晶のスタティック駆動回路



電圧と等しくなります ($5V_{rms}$)。なお、TTL で液晶を駆動すると“H”レベルの不安定さのためうまくいきませんので念のため。

図 4.25 (b) に示すのは 7 セグメントの数字表示用のものです。この IC にはラッチ、デコーダ、Ex-OR ゲートが共存しているので、CPU のバスへの接続が容易です。ただしバスの“H”レベル電圧とホールド時間とに注意します。

桁数が多くなると駆動も大変になってきます。同図(c)は 7 セグメント \times 4 桁の駆動用 IC です。この IC も“H”レベルに注意すれば、CPU に直結できます。桁数を拡張する場合、液晶の背面電極（バック・プレーン）が共通のものは同じ位相の信号を印加しなければなりません。図(c)の IC では内蔵の発振器間の同期をとることができます（同図(d)）。

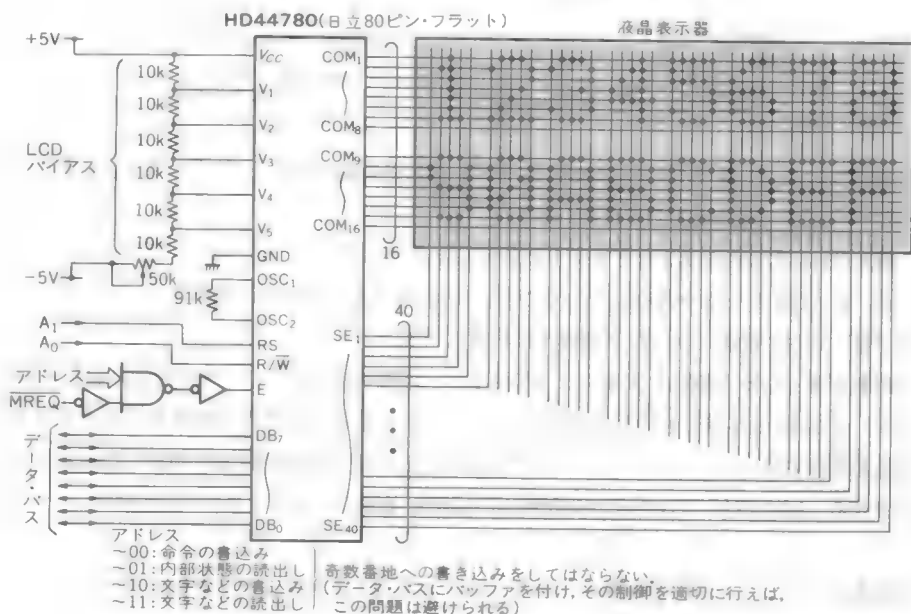
▶ **液晶のダイナミック駆動回路** さて、以上はすべてスタティック駆動の例でしたが、これ以上のセグメント数となると端子数だけでも大変なのでダイナミック駆動を考えます。

液晶のダイナミック駆動についてはいくつかの方法がありますが、前述のように直流印加が不可能な点と応答の遅さとから、駆動はかなり大変です。そこで、ここでは専用のICの利用を紹介します。

図4.26は5×7ドット・マトリクスの文字表示素子を駆動するコントローラです。このコントローラHD44780⁽¹²⁾はマイコン側からデータやコマンドを受け取り、内部の文字発生ROMを参照し、液晶を駆動します。端子数の関係から、このICだけでは16文字(5×8ドット×16文字)までの駆動しかできませんので、桁数を増す場合には駆動部分だけのIC(HD44100など)を追加します。HD44780からHD44100へは液晶駆動用の直列信号が送られます(なお、HD44780はフラット・パッケージです)。

液晶表示器自身は比較的容易に大型のものが作れますが、このように駆動が大変なため普及が遅れています。しかし最近では、白黒テレビに匹敵するドット数(2万ドット以上)のものが駆動用IC付きで入手できるようになってきたので、今後はもっと使用されるようになるでしょう。

図4.26 液晶のダイナミック駆動回路



例 4-11 リレーの駆動

マイコンから高電圧・大電流の負荷をオン/オフしようとするとき、もっとも一般的なのはリレーを使うことです。ただし、リレーは誘導性負荷であり、駆動には注意が必要です。

一方、最近ではラッチング型など各種のリレーが使われるようになってきました。これらの使い方についても触れることにします。

▶**リレー駆動のための条件** 出力インターフェースから見ると、リレーは誘導性負荷であって、ある程度電力も必要とします。そのため、つぎの項目を満足する必要があります。

- (1) コイルに必要な電力を供給できること。
- (2) 必要な速度で電流をオン/オフできること。
- (3) 電流オフの際に生じるサージ電圧を吸収すること。

一方、出力としてのリレー接点には何らかの電気回路が接続されるわけで、そちら側からはつぎのような点が要求されます。

- (4) 電力用としては、十分な接点容量（電圧、電流）をもつこと。
- (5) 信号用としては安定な接点接触を保つこと（接触不良、チャタリング、接触電位などを含む）。
- (6) 必要な速度での開閉が可能なこと。

普通、(4)、(5)、(6)から、まずリレーを選びます。

$$\frac{\text{リレーの接点で開閉できる電力}}{\text{コイルを駆動するのに必要な電力} \times \text{容積 (cm}^3\text{)}}$$

をリレー効率と呼びます⁽¹⁵⁾が、最近ではきわめて高感度のリレーが現れ、リレー効率も1000以上におよんでいます。また、外形寸法も小型化され、125V 2Aの接点容量をもっているも16ピンDIPのICと同じくらいの平面形で間に合います。

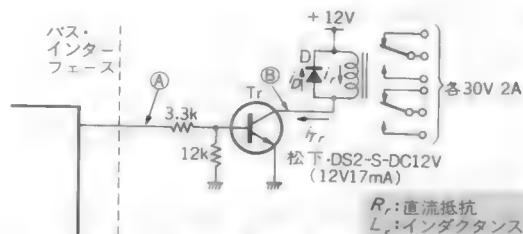
一方、駆動回路を内蔵したリレーや、有極リレーにICを組み込んだ高速・高感度のリレーなども入手でき、リレーの選択も駆動回路の設計も、従来よりはるかに楽になってきました。

▶ **リレーの基本的駆動回路** 図4.27(a)に示すのは、30V 2 Aの負荷を接続できるリレー駆動用インターフェースです。リレーは誘導性であり、そのままでは電流をオフした瞬間に高電圧を発生します。図中のダイオードDはそれを防ぐためのものです。同図(a)のTrをオフにすると、それまでリレーに流れていた電流は、そのままDに流れますから、ダイオードはそれに見合った順方向定格のものを使用します。

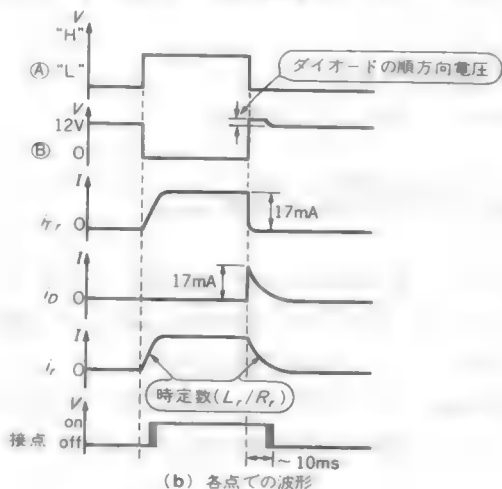
なお、同図(a)のようにたんに逆並列にダイオードを付加した場合、リレーをオフにしようとしても時定数 L_r/R_r でしか電流が減衰しません。

これに対してオフ時間を速くするには、図4.28のような回路を使用します。同図(a)はダイオードに直列に抵抗 R_1 を挿入し、時定数を $L_r/(R_r+R_1)$ としたもの、同図(b)は非線形素子（ここでは定電圧ダイオード）を挿入したものです。いずれもオフ時間を図4.27の数

図4.27 基本的なリレー駆動回路



(a) 駆動回路例



(b) 各点での波形

分の1にすることができます。

▶ **リレーの駆動回路** 図 4.29 に示すのは微小信号負荷用としてリード・リレーを用いたものです。リード・リレーは微小信号に対しても安定ですが、接点の熱起電力が問題になるような箇所では発熱や接点間容量にも注意が必要です。また、リード・リレーの高速性を利用したい場合（リード・リレーは 100Hz での開閉も可能）には駆動回路もそれなりに速度に注意します。

図 4.30 は有極（2巻線ラッチング）リレーの駆動回路です。以前は有極リレーは特殊な用途にしか用いられませんでした。最近では新たに高性能の小型有極リレーが開発されています。これらのリレーは高感度、高速で、常時電流を流しておく必要がないため低消費電力であり、自己保持作用があることから様々な用途が考えられます。

図 4.30 (a) は二つのコイル（巻線）を差動的に駆動する例ですが、CR を直列に挿入してあるので切替の瞬間にはコンデンサの充電電流が短時間だけ流れます。定常状態では R

図 4.28 オフ時間を速くする

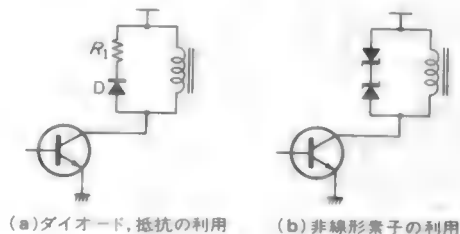
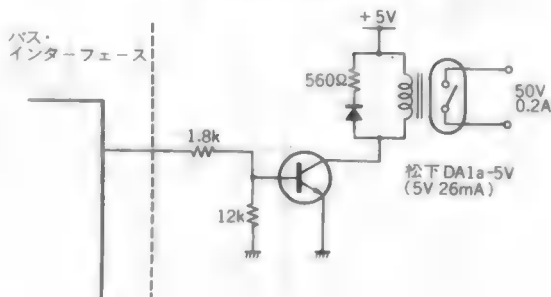


図 4.29 リード・リレーの駆動



を通してわずかの電流が流れるだけです。この回路ではオン時に十分な電流・時間積が得られるように CR を選ぶことが大切です。一方、 C が充電してしまわないと再切換えが不可能ですから、あまり早い周期でのオン/オフはできません。また同図(b)は、ワンショット回路を使ったもので、④点のレベルにしたがってリレーがオン/オフします。いずれの場合も、ノイズでリレーがオン/オフしないように注意が必要です。

図 4.30 ラッチング型リレーの駆動

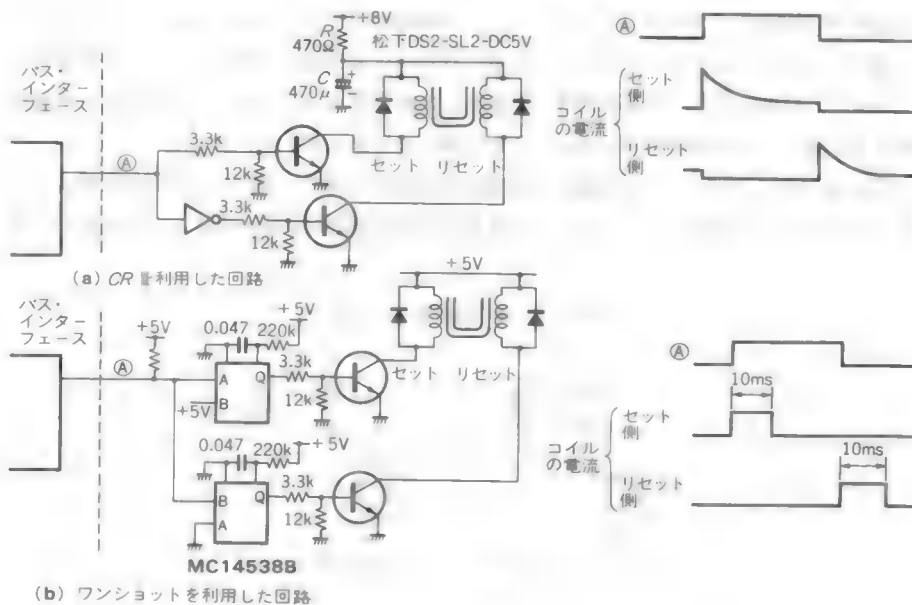


図 4.31 パワー・リレーの駆動

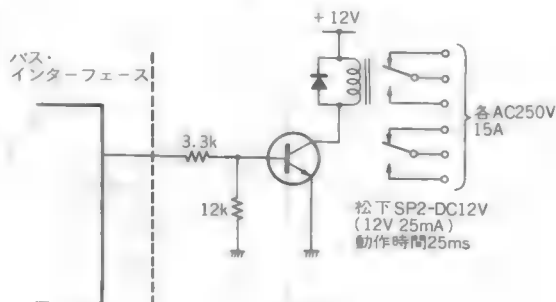
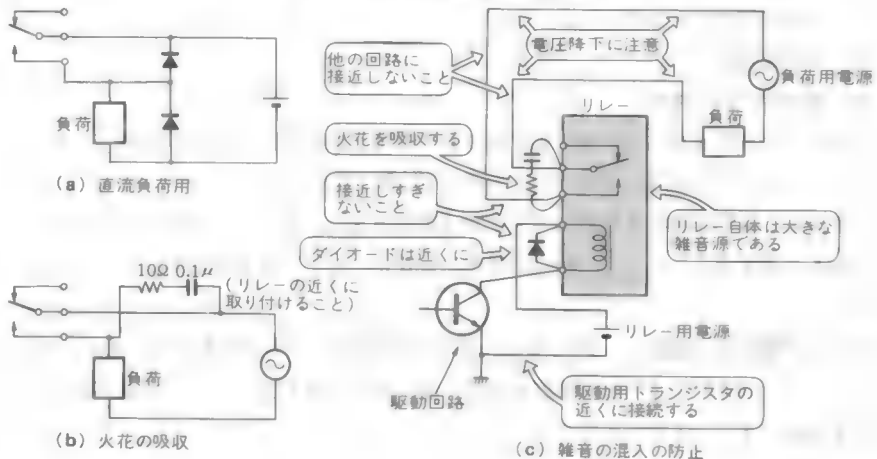


図 4.32 接点の保護と雑音の防止



▶ **パワー・リレーの駆動回路上の注意** 図 4.31 に示すのは 250V 15A 級のパワー・リレーの例です。パワー・リレーになると駆動電流も多く必要かわり、速度はあまり必要ではありません。ただし、大電流・高電圧の開閉による雑音などがマイコン側に影響を与えないように考慮しなければなりません。

リレーの負荷が直流の場合、図 4.32 (a) のようなダイオードにより、リレー接点を保護するとともに、火花による電磁波の発生を抑えます。同図 (b) は交流でも直流でも効果があります。

リレー自体は技術の進歩により小型化され、パワー・リレーであっても基板上に載るようになりましたが、逆にパワー関係の配線が他の回路に近接して新たな問題が生じています。図 4.32 (c) のような点を考慮して、インターフェースを介して雑音が混入するようなことがないように注意すべきです。

例 4-12 アナログ信号の出力①——アナログ・スイッチによる方法

マイコンからアナログ信号を出力するには、一般には D-A 変換器を使用します。ただし、必要とする分解能が低くてよい場合、いろいろな手法が使えます。

ここでは、6 ビット以下の簡単なアナログ出力インターフェースの例をあげ、タイミングや絶縁についての考え方を説明します。

アナログ信号にも様々な種類がありますが、やはりつぎのような点に注目します。

- (1) 信号の形態とレベル
- (2) 必要な精度
- (3) 必要な変化の速度

上記のうち、(1)と(2)とは付加する回路の工夫で何とかできますが、例によって(3)が一定になると、CPUでは対応できません。たとえば、図4.33のようなファンクション・ジェネレータでは、出力周波数は100Hzオータが上限でしょう。速いほうの例にはビデオ信号の出力がありますが、これには一種のDMAを必要とします(第5章参照)。

▶アナログ電圧出力回路 図4.34に示すのは数段階のアナログ電圧出力用インターフェースです。いずれもアナログ的に設定した電圧を切り換えるもので、同図(a)は2段階、(b)は8段階(3ビット)のものです。

図4.33 プログラマブルなファンクション・ジェネレータ

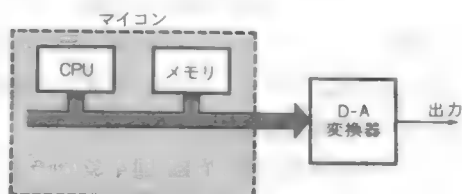
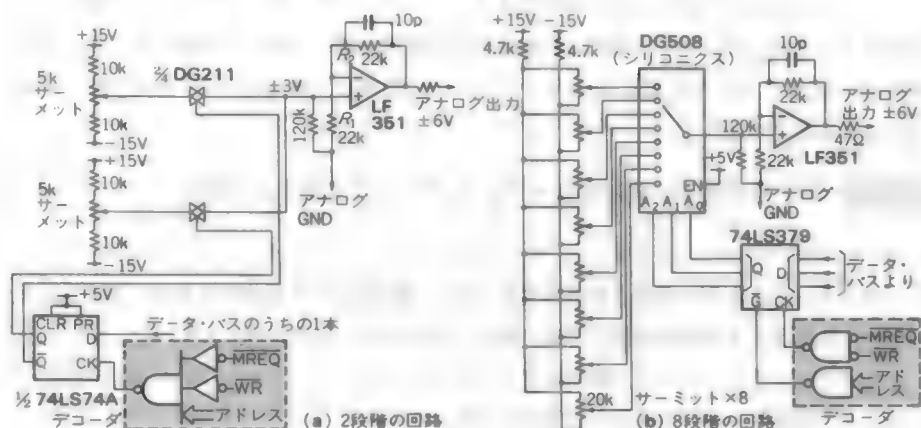


図4.34 数段階のアナログ出力用インターフェース



アナログ・スイッチは微小信号に対しても接触不良やチャタリングの心配がなく、速度も $1\mu\text{s}$ 前後と高速です。ただし、オン抵抗やオフ時の漏れ電流が存在します。同図(b)の DG 508⁽¹⁴⁾ でも最大数 100Ω のオン抵抗があるので、高インピーダンスで受けるようにします。

もう一つの問題は、切換え時のタイミングです。DG 508 などのアナログ・マルチプレクサの多くは BBM (第3章コラム H 参照) 動作です。両方のスイッチがオフになっている期間は数 100ns で、通常は問題ありませんが、とくに高速・低雑音が要求される場合は図 4.35 のように過渡状態にも注意が必要です。

一方、MBB のスイッチでは、スイッチ間のショートに注意しなければなりません。過渡状態では中間の電位が出力されます。

▶ アナログ出力をマイコンから直接導出する回路 つぎに、アナログ信号をマイコン側と絶縁したい場合の例を図 4.36 に示します。

図 4.36 (a) はアナログ・スイッチのかわりにリレーを用いたもので、もっとも簡単です。ただし、リレーには接触不良、チャタリング、動作時音などの問題があります。図 4.36 (a) の DS 型リレーでは、最小適用負荷が $10\mu\text{A}$ 10mV (DC) となっています⁽¹⁵⁾。しかし、オン

図 4.35 スイッチの切換えタイミング上の注意

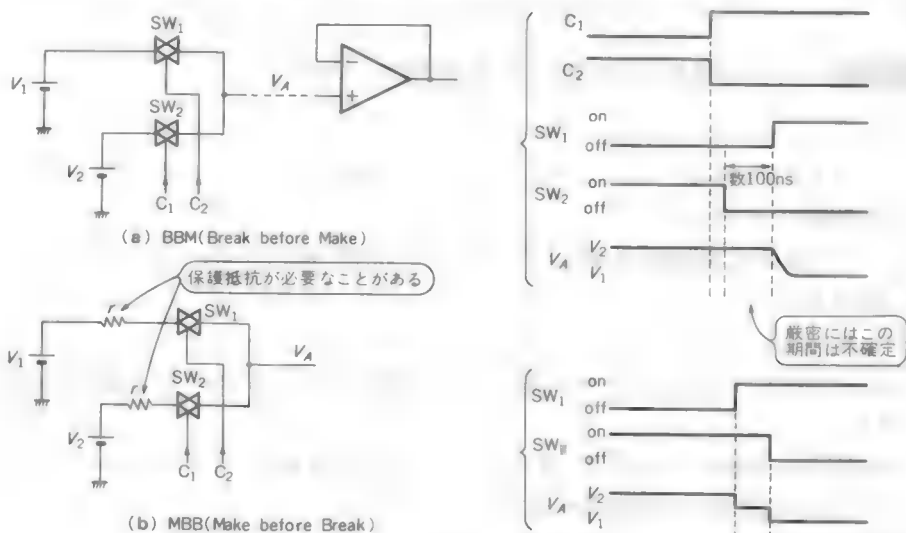
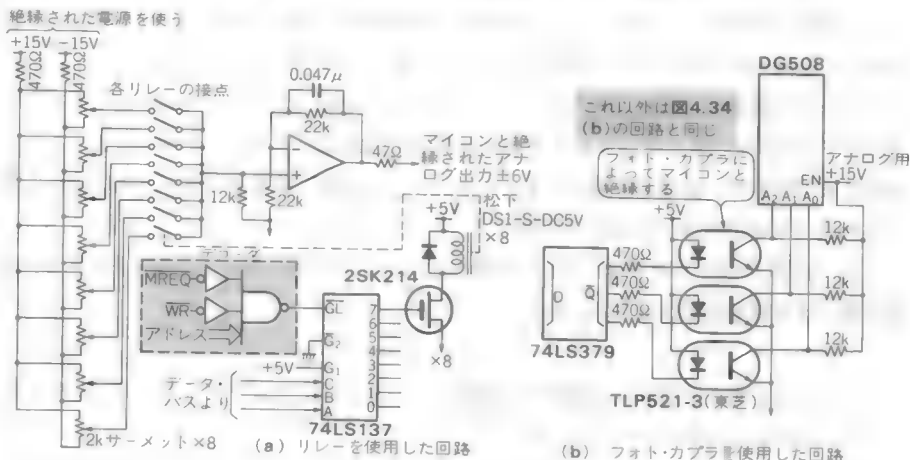


図 4.36 絶縁されたアナログ出力用インターフェース



抵抗はアナログ・スイッチと違って数 $10\text{m}\Omega$ ですので、安定な開閉を望むには低インピーダンスで受けることです。この程度の小型リレーでは動作時間は数 ms 、リード・リレーでは 1ms 以下となります。チャタリングなどはこの時間内に収まると見て良いでしょう。

リレーでは動作時間が遅すぎる場合、同図(b)のようにフォト・カプラでの絶縁を考えます。フォト・カプラでとくに高速型と断っていないものでは、数 μs の遅れがあります。

例 4-13 アナログ信号の出力②——D-A 変換器による方法

アナログ信号の出力において、6ビット以上の分解能が必要ならば本物(?)のD-A変換器を使用します。

ここではD-A変換器の基本的な使い方と、高分解能・高精度を得るための注意とを述べます。

各ビットのデジタル・データに比例した電圧や電流を出力するには、D-A変換器を用います。

市販のD-A変換器のなかには、 $R-2R$ ラダー(コラムM参照)およびアナログ・スイッチだけから成るもの、基準電源や出力バッファを含んだもの、バス・インターフェース回路を含んだものなど、多くの種類があります。

図 4.37 簡単な D-A 変換器

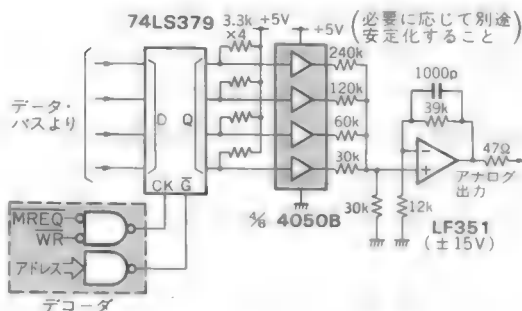
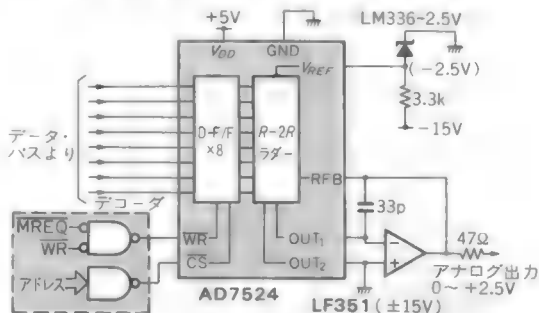


図 4.38 分解能 8 ビットのアナログ出力



■ CMOS を利用した A-D 変換回路 図 4.37 に示すのは非常に簡単な例です。CMOS のデジタル IC は出力が電源電圧いっぱいに振れるので、このような使用が可能です。ただし、出力インピーダンス（5V 時に数 100Ω）の制限から、6 ビットが限界でしょう。

▶ D-A 変換器を使用した回路 これ以上のビット数では本物（？）の D-A 変換器を使います。図 4.38 は 8 ビットの例です。アナログ・デバイセ社の AD7524⁽⁶⁾ は、AD7523 に入力ラッチを設けたものと考えればよく、マイコン・バスへの接続が容易です。ただし、電源電圧が 5V のときにしか TTL レベルとはならない点も 7523 のままなので注意が必要です。

ビット数がデータ・バスの幅（たとえば 8 ビット）を越える場合、データを同時に印加できるよう注意が必要です（例 4-4 を参照）。

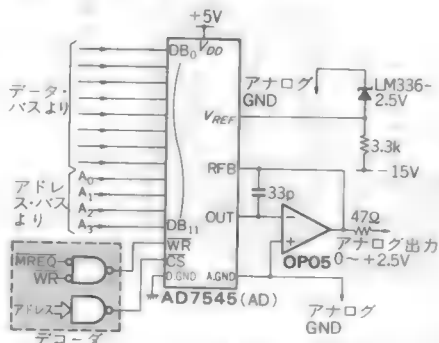
図4.39に示すのは12ビットの例で、データ・バスを2回使うか、あるいはアドレスを使用するなどして、12ビットを同時に出力するよう努めます。

図4.39(a)のAD7545は、内部に12ビットのデータ・ラッチ（エッジ・トリガのDフリップフロップ）をもっているため、外部回路は簡単です。

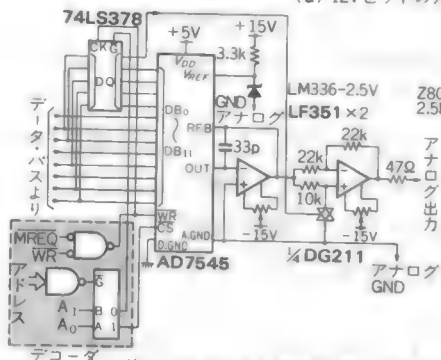
同図(b)は同時に符号の切り替えも行っていますが、符号切り替え用のスイッチとD-A変換器との速度の違いが大きい場合（たとえばアナログ・スイッチのかわりにリレーを使用する場合）、アナログ出力に大きなグリッチを生じさせることになるので注意しなければなりません。

同図(c)は内部に2段のラッチをもつICの例です。ただし、このICはデータ・ホールド時間が90ns必要で、タイミング計算には手が抜けません。

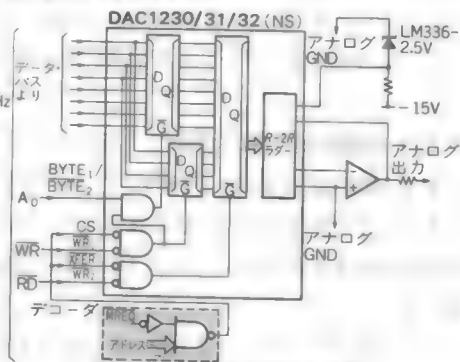
図4.39 分解能12ビットのアナログ出力



(a) 12Vビットのアナログ出力インターフェース



(b) (12ビット+符号)のアナログ出力



(c) 2段ラッチを含んだD-A変換器

▶ **必要な精度を得るための注意点** さて、技術の進歩により最近では16ビットのD-A変換器でも容易に入手できるようになってきました。しかし、D-A変換器が16ビットであっても、アナログ出力としてそれに相当する精度（広義の）が得られるわけではありません。そこで、実際によく使われる12ビット程度のD-A変換器で、必要な精度を得るための注意点をあげておきます。

(1) **基準電圧源の温度特性** D-A変換器によって基準電圧源を内蔵したものと外付けのものがありますが、いずれにしても温度特性には十分な注意が必要です（例3-14参照）。

(2) **アナログ・グラウンドを統一する** デジタル・グラウンドとアナログ・グラウン

図 4.40 アナログ出力用インターフェースの構成例

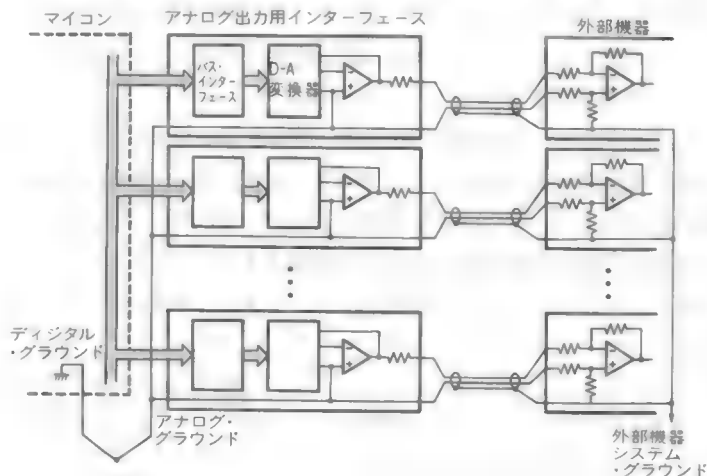
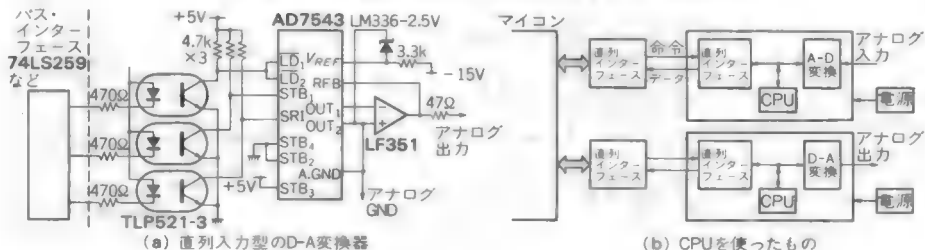


図 4.41 直列転送を用いたアナログ出力用インターフェース



ドを分離することはもちろん必要です。ただし、それだけでなく、各アナログ出力信号は外部機器側で差動で受けてもらうのが望ましいといえます(図4.40)。

とくに、1台のマイコンにA-DやD-Aが何系統もふら下があると、グラウンド間の電位差があちこちで生じます。これが許容できない場合は思いきって絶縁を考えます。その場合、バス全部をフォト・カブラで切ると本数が多いので、速度が遅くてよければ直列転送を考えます。図4.41(a)はデータを直列に入力するD-A変換器の例です。RS-232Cなどで結んだ図4.41(b)の構成も、マイコンが安くなった現在、決して実現困難ではありません。

コラム1 Dフリップフロップとトランスパレント・ラッチ

ストロブ信号によるデータのラッチ(固定)や、時分割バスからのアドレスの分離、あるいはデータの出力用として、Dフリップフロップやトランスパレント(透過型)・ラッチが使われます。たとえば、74LS373はトランスパレント・ラッチ、74LS374はDフリップフロップです。

これらは大体同じような用途に用いられ、用途によってはどちらでも使用可能な場合もあるため混同または誤用することがありますが、本当は両者は明確に異なります。

図1.1(a)はDフリップフロップ、(b)はトランスパレント・ラッチの動作です。前者がクロック信号CKの立上り時点のデータを保存するのに対して、トランスパレント・ラッチはゲート信号Gが“H”の間は入力データをそのまま出力します。トランスパ

図1.1 Dフリップフロップとトランスパレント・ラッチとの動作

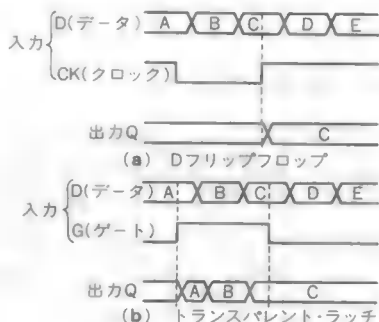
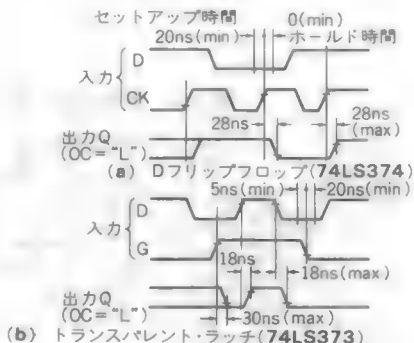


図1.2 74LS374と74LS373とのタイミングの違い



レント(素通し)の名はここからついています。また図1.2に示すように、必要とするセットアップ時間やホールド時間が異なります。

つぎに、具体的な使用法を示します。図1.3は時分割バスからアドレスを分離するアドレス・ラッチ回路です。ここには、Dフリップフロップでもトランスバレント・ラ

図1.3 時分割バスからのアドレス分離回路

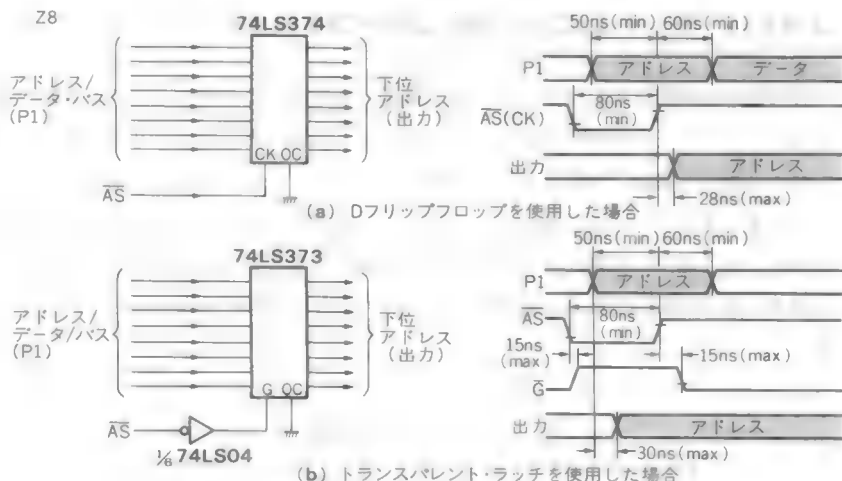
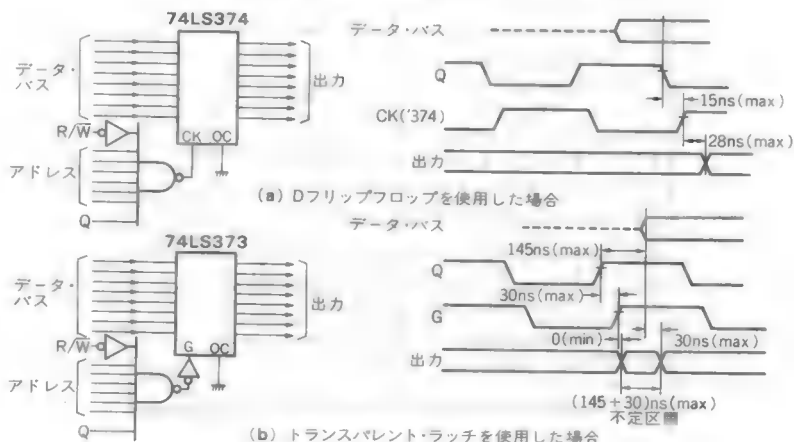


図1.4 6809 出力インターフェース回路



ッチでも使用可能ですが、後者のほうがアドレスが早く出力されます。

図1.4に示すのは6809の出力インターフェースです。第3章で示したように6809はE（データ・ストローブ）の立上り時点ではデータが確定していません。したがって、ここにトランスバレント・ラッチを使うと、データの書換え時にデータ不確定の期間を生じることがあります。

コラムJ セントロニクス型インターフェース

片方向の並列信号をハンドシェイクで転送するには、このセントロニクス型と呼ばれるインターフェースが代表的なものとなっています。とくに、プリンタやブロックでは大部分のものが“セントロ”を自称しています。

いうまでもなく、この言葉はセントロニクス社製のプリンタの仕様との互換性を示します。ところが、実際には元祖との完全互換性のあるものはまれで、多くの製品は仕様の一部が共通しているに過ぎません。このことをつぎの4点に分けて考えてみます。

表 J.1 タイミングの分類

製品	ノーク	タイミングの仕様(単位 μs 時記号のないものは代数值)							備考
		分脈	①	②	③	④	⑤	⑥	
A	1	A	0.5 以上	0.5~200	0.5 以上		250 以上	12 以下	⑦=17 以上
B	2	A	1 以上	1 以上	1 以上			4 以上	
C	3	A	1 以上	2 以上	1 以上	100 以下	800	40	
D	4	A	0.5 以上		0 以上			2~10	
E	5	A	0 以上	0.5 以上		0 以上	200 μs ~ 200 ms		①=0.5 以下 ②=0.5 以上
F	6	B	1 以上	1~100	③=1 以上	④=30 以上	60 以上	30	⑦=30, ⑧=0 以上
G		A	1 以上	1~100	③=1 以上	④=0~0.5	100 以上	5~10	⑧=0 以上
H	7	C	1 以上						⑨=2 以上
I	8	B		0.5~2				2	
J	9	B	1 以上	1 以上	1 以上	④=100 以下		6~8	
K	1	A	0.5 以上	0.5 以上			64~100	154 以下	
L	10	A	0.5 以上	0.5 以上	0.5 以上			6	⑩=1000 以下 ⑪=0 以上

(1) コネクタやケーブルなどの接続仕様

過半数の機器はセントロニクス社と同じ DDK36P のコネクタを採用しています。しかし、勝手にコネクタを付けても“セントロ”を自称しているものがあります。

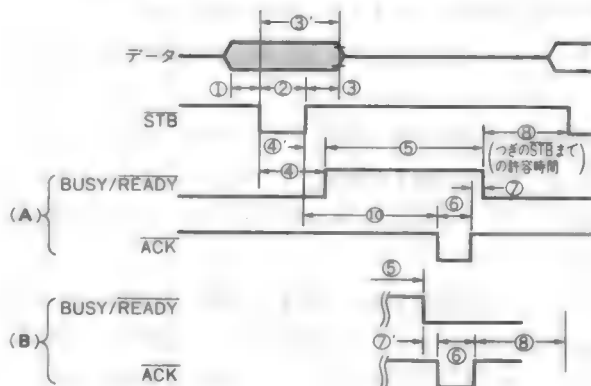
ケーブルはすべてツイスト線を使うことになっていますが、実際には平行フラット・ケーブルが使われることが多いようです。短距離ならばこれでもかまいませんが、望ましくはありません。

しかし、コネクタやケーブルについては、違いが目に見えるものだけに、付属のものをを使う覚悟さえあれば、問題となることは少ないといえます。

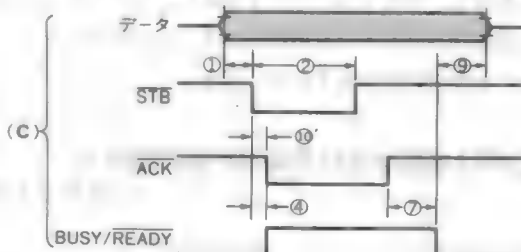
(2) 信号線の種類

データ（8ビット）とハンドシェイク用の信号（3本）とだけではなく、セントロニクス社のものにはリセット、紙送り、紙切れなどを示す信号が何本か付属していま

図 J.1 プリンタなどのタイミング例



(A):BUSY/READYの立下りよりもACKの立下りのほうが早いもの
(B):BUSY/READYの立下りよりもACKの立下りのほうが遅いもの



す。これらのうち付属の信号については完全に同じであるものが少数派です。しかし、これもとくに問題とはなっていないようです。

(3) 制御用の符号について

印字、改行、改ページ、倍幅文字などの指令には、ASCIIのうち制御コード (00H ~ 1FH) を使用しますが、これもまた、機器により違い、これについてはぜひとも ISO や CCITT の取り決めにしたがった適切な使い方を定めてほしいと思います。ただし、これも主としてソフトウェア上の問題ですから、本書では深入りしないことにします。

(4) 信号線の電氣的仕様およびタイミング

本コラムで詳しく取り扱いたいのはこの点です。とくに重要なデータ線および3本のハンドシェイク線の直流特性とタイミングとに限って調べてみます。

まず、外部機器の入力側 (データおよび $\overline{\text{STB}}$) の直流特性では、シュミット入力バッファに GPIB 並の終端を施してあるものから、コントローラ (8041 や 8048 系、あるいは 3870 などの F8 系が多い) の MOS 入力に $10\text{k}\Omega$ くらいの軽いプルアップ抵抗をつけただけのものまであります。同一筐体への組込み用のものを別にして、ちゃんとした終端をつけるとともに送り出し側も十分な駆動能力をもたせることをおすすめします。

つぎに外部機器からの出力側 ($\overline{\text{ACK}}$ および $\text{BUSY}/\overline{\text{READY}}$) については、ひどいものはコントローラの出力線を直接出しているものもありますが、これではマトモな終端は不可能です。できれば $I_{OL} \geq 16\text{ mA}$ ($V_{OL} = 0.4\text{ V}$) 以上の駆動能力がほしいところです。

最後に、タイミングの問題を調べてみます。■ J.1, 図 J.1 に示すのは手持ちの資料のなかからプリンタやプロッタの入力タイミングを集めたものです。

これだけバラバラのものを“セントロ”の一言ですませってしまったことにも感心しますが、なかには取扱説明書にも仕様書にもタイミングについての記載が存在しない製品もあり、恐ろしくて使う気になれません。

今後は GPIB (IEEE-488) との互換性のあるものが増えてくると思いますが、当分は“セントロ”の文字に振り回されそうです。

コラムK CPUを選ばない万能 I/F は可能か ①

——8ビット汎用 CPU

家電用の4ビットのシングルチップ・コンピュータから、メインフレーム用の32ビ

ットCPUまで、すべてに接続できるものを作るのは現実的ではありません。しかし、現在、制御用やパーソナル・コンピュータ用として使われている何種類かの代表的CPUについては、どのCPUにでも接続できるようなインターフェースを考慮しておけば非常に便利です。まず手始めに、8ビットの汎用CPU（Z80、6809、8088など）について考えてみます。

●データとアドレスとは多重化されているか？

上記3種の中では8088が時分割バスとなっています。このほか、8048やZ8などのシングルチップ機はたいてい時分割バスです。COSMAC（RCA、1802系）はアドレスだけの変則時分割バスです。

ただし、これはアドレス・ラッチを設けてアドレス・バスを分離しておけば、たいした問題ではありません。

■アドレス空間の違い

6809は64Kバイトの単一アドレス空間、Z80やZ8はメモリ領域とI/O領域に分かれた128Kバイト、8088（ミニマム・モード）はメモリ領域1MバイトおよびI/O領域64Kバイトなど、CPUによってアドレス空間が異なります。

ここで非常に残念なのは、Z80のメモリ領域とI/O領域とを区別する信号が、他のアドレス信号と異なるタイミングで出力される点です（後述）。

図K.1 共通インターフェース——その1

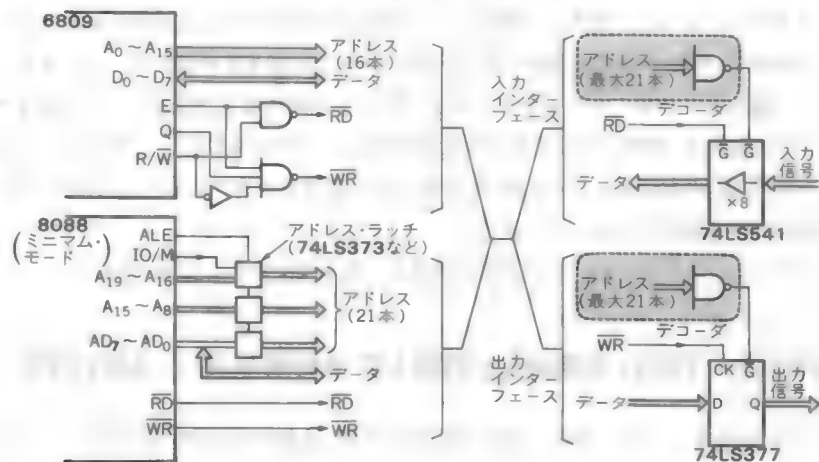
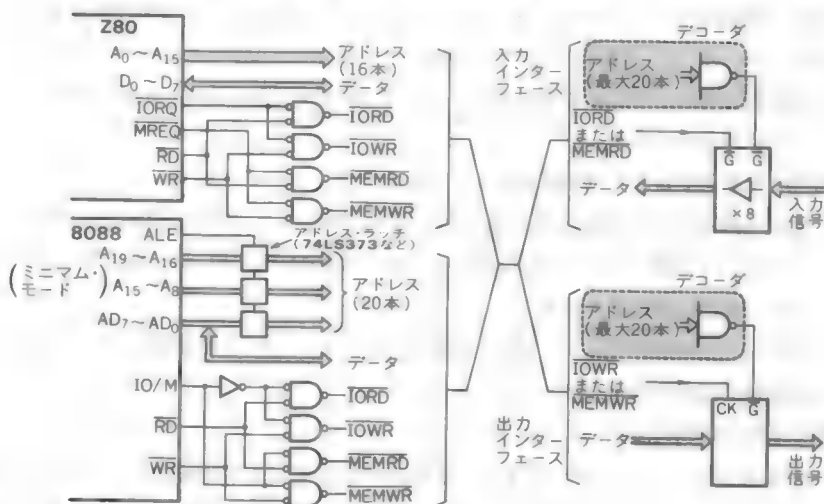


図 K.2 共通インターフェース——その2



●データ・ストロブ (\overline{RD} , \overline{WR}) の出方の違い

6809はE(およびQ)および R/\overline{W} 、Z80や8088は \overline{RD} および \overline{WR} で、これは互いに変換可能です。他のCPUの多くも、このいずれかに分類できます。

ところが、先程の問題がかかわってきて話が面倒になっています。すなわち、メモリ領域とI/O領域を区別する信号を、アドレス線とみなすか、データ・ストロブ線と見なすかによって、図K.1、図K.2の2種類の形式に分けざるを得ません。

ちなみに、8080の時代に登場したS-100バスなどは図K.2の形になっています。しかし、統一アーキテクチャを考えると、図K.1の形のほうが望ましいと思えます。

さて、図K.1、図K.2のような簡単な変換を行った場合のタイミングですが、LSタイプのTTLやHCタイプのCMOSを使った入出力インターフェースに対しては遅延的にもある程度の余裕があります。

ただし、メモリはタイミング的に相当厳しくなりますから、注意が必要です。

コラムL CPUを選ばない万能I/Fは可能か②——MOTEL

古くは8080A対MC6800、最近の高性能8ビット機では8088対6809、シングルチップ分野では8048対6805など、インテル社とモトローラ社はライバル関係を続け

図 L.1 制御信号の違い

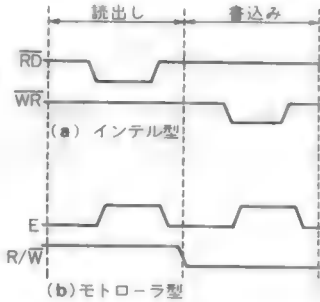


図 L.2 MC 146818 の MOTEL インターフェース部分

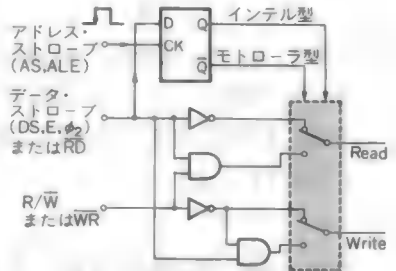
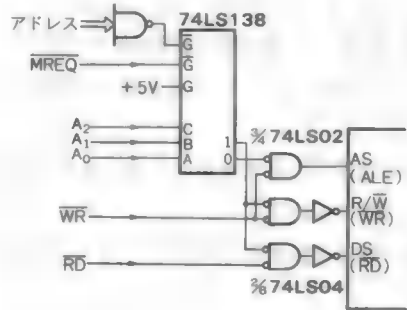


図 L.3 時分割バス用の IC を分離バスに接続する 1 例



ています。

ハードウェアの観点からは、インテル社が \overline{RD} と \overline{WR} との 2 本のデータ・ストロブ信号を使っているのに対して、モトローラ社は E (古くは ϕ_2) と R/ \bar{W} を使っています。これはシングルチップ機から 32 ビット機に至るまですべて同じです。

この両者を自動的に判別してしまおうというのが、“MOTEL” (Motorola-Intel) と呼ばれるインターフェースです。

図 L.1 に両者の信号を示しますが、バス・サイクルの開始点に注目すると、モトローラ型の E 信号が必ず“L”であるのに対して、インテル型の \overline{RD} 、 \overline{WR} はともに必ず“H”であることがわかります。

図 L.2 に示すのは MC 146818 の MOTEL インターフェース部分です。この IC はバス・サイクルの開始を判定するのに ALE (アドレス・ストロブ) を使っているので、データとアドレスとが分離されている 6809 や Z 80 には直結できません。しかし、6805 と

や 8048 などのほか、8085、8088、Z8 などにも接続可能で、使用できる範囲はかなり広いといえます。

*

*

*

さて、せっかく時分割バスに直結できるように作られた IC なのですが、それを 6809 や Z80 に接続したいという質問が必ず出てきます。それに対するもっとも簡単な方法を図 L.3 に示します。CPU からは必要なアドレスを出力してからデータの読み書きを行うようにします。

コラムM R-2R ラダーと D-A 変換器

従来、D-A 変換方式として主流を占めていたのは、電流加算型と R-2R ラダー型です(図 M.1)。乗算型(基準電圧に比例した出力が出る)D-A 変換器の場合、R-2R 型のほうが使いやすいこともあり、最近の D-A 変換器では R-2R 型を多く見かけます。

R-2R 型には電流出力と電圧出力との 2 通りの使い方があります(図 M.2)。いずれも OP アンプによってインピーダンス変換を行う必要がありますが、図に示すようにその接続方法が異なります。とくに、電流型では基準電圧と OP アンプ出力との極性が

図 M.1 D-A 変換方式

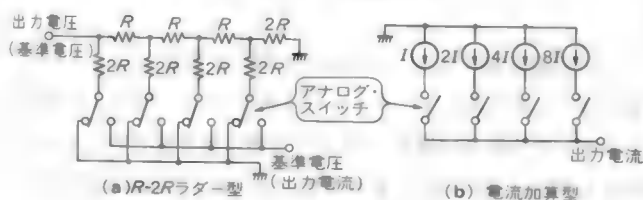


図 M.2 R-2R ラダーの 2 通りの使い方

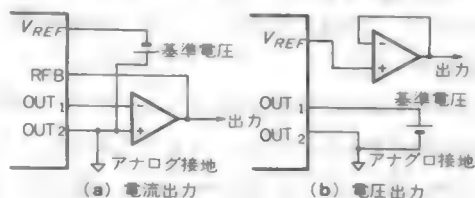
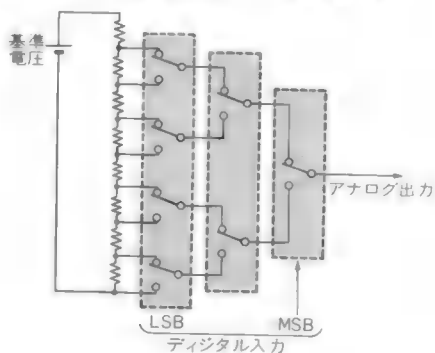


図 M.3 比例分割型 D-A 変換器の原理



逆になる点に注意して下さい。最近のCMOS構造のA-D変換器はどちらの使い方もできるものが多いのですが、内部のアナログ・スイッチの構造によっては、電流出力でしか使えないものもあります。

一方、最近のD-A変換器やA-D変換器(逐次比較型では内部にD-A変換器を持っている)のなかには、従来の R - $2R$ 型とは異なるものが現れました。図M.3に示したのはその例で、原理的には非常に簡単です。ただし、抵抗やアナログ・スイッチが多量に必要ですが、IC技術の進歩により、問題となってはいません。

これ以外にも、 R - $2R$ 型と同じ構成で R - nR ($n \neq 2$)型としたものがあります。このような回路でD-A変換を行うと、デジタル入力値とアナログ出力値とは比例しませんが、入力信号をROMに印加して値を変換して問題を解決しています。

第5章

高度な入出力インターフェースの設計

前章までは基本的な入力および出力インターフェースについて述べました。しかし、よく用いられるインターフェースのなかにも、■3章または第4章に書ききれないものが少なくありません。それらは、つぎのような理由によるものです。

- (1) 映像信号のように、マイコンのソフトウェアでは対応しきれない高速のデータ転送を必要とするもの。通常、このような場合はDMA（直接メモリ・アクセス）を用いる。
- (2) IEEE-488 (GPIB) のように、専用のインターフェース用 LSI を用いることにより、容易に所定の仕様を実現できるもの。IEEE-488 のほか、同期・非同期の直列通信回線 (RS-232C など) もこれに含まれる。

以上のうち、映像信号は市販マイコンの標準的な出力として、また IEEE-488 や RS-232C は標準的な入出力として、ほとんどのものに装備されています。このほか、今後普及しそうな高速 LAN（ローカル・エリア・ネットワーク；Ethernet はその代表的なもの）は、

(1)、(2)の両方に該当します。また、音声認識や音声合成もこの範ちゅうに入ります(図5.1)。

これらの処理のなかには以前はマイコンではまったく不可能なものもありましたが、マイコンのハード、ソフト両面の進歩により、あるいはアルゴリズム（手法）の進歩により、着実にマイコンで処理できる範囲が広がりつつあります。また、図5.1の範囲をさらに越える高次元の配列演算なども、机の上に乗るようなシステムで実現できるようになると思われます。

さて、余談はさておき、このような（ソフトウェアだけでは対抗できない）処理を行うには、つぎのような点を考慮する必要があります。

(1) 高速の処理には DMA

汎用の CPU (Z80 など) では、外部信号に対してもっとも速く応答できるのは DMA で

図 5.1 高度な入出力インターフェースの例

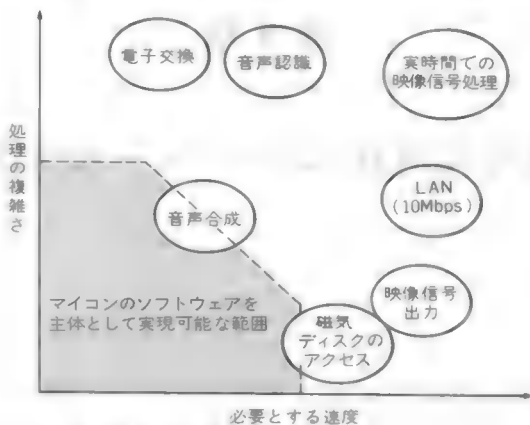


図 5.2 ソフトウェア制御による直列通信

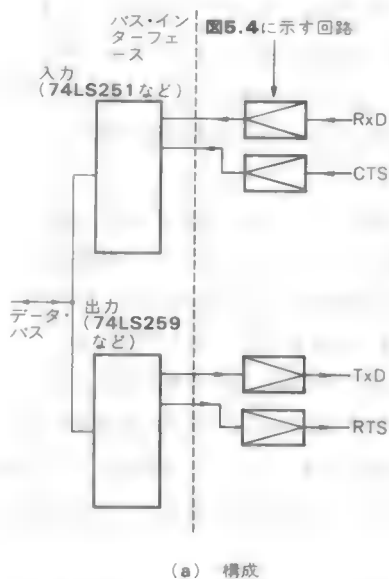
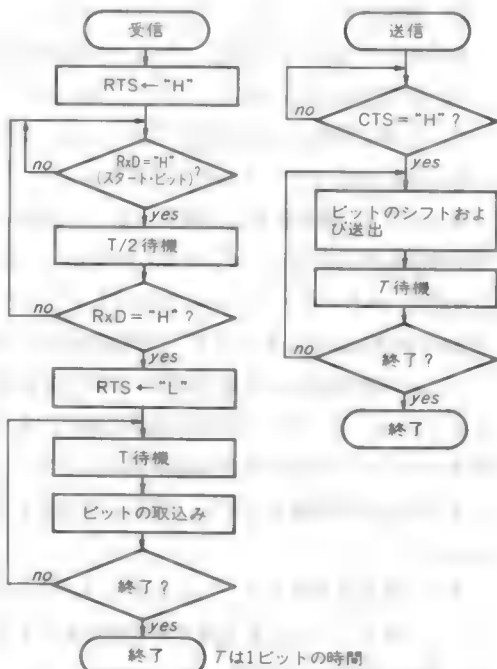


図 5.4 に示す回路



(RS-232C のターミナル型の場合)

(b) ソフトウェアの概要

す。ただし、サイクル・スチール^{*}としない場合、CPUが行う通常の作業はストップします。

(2) 複雑な処理には専用 IC を使う

上記 DMA の制御のほか、各種の通信、映像信号の入出力、音声の認識や合成などの仕事には、専用の LSI が発表されています。可能でさえあれば、これらの IC を使うのが得策です。ただし、これらの IC を CPU に接続するには、両者の仕様を十分に検討する必要があります。

(3) 分散処理・並列処理を考える

どうしても処理が間に合わなければ、複数 CPU を使って分散処理または並列処理を行います（もっとも、前項の専用 IC の内部は、多くの場合シングルチップ・コンピュータとなっており、広い意味では分散処理といえる）。この場合、CPU 間のデータ転送方法が決め手となります。

本章では、このような入出力インターフェースからいくつか例をあげて説明します。

5.1 直列通信回線のインターフェース

例 5.1 直列通信回線

直列通信回線にも“たれ流し（ハンドシェイクを行わない）”式の低速のものから、10Mbps を越える LAN（ローカル・エリア・ネットワーク）までありますが、ここではマイコンに装備されることの多い非同期（または調歩同期）式のものを考えてみます。

図 5.2 に示すのは直列非同期通信をソフトウェアで行おうとするものです。図の例ではデータ（ T_xD , R_xD ）と CTS および RTS とだけを扱っていますが、これ以外の制御信号が必要ならば、それらもソフトウェアで処理します。同図(b)にソフトウェアの概要を示しますが、Z80A を 4 MHz で使用した場合、このような半二重的な使い方では 4800bps くらいまでは十分に実用になります。ただし、送受信を行っている間、CPU はそれに専念する必

* サイクル・スチールとは、CPU が行うバス・サイクルの■をぬって、外部機器から DMA を行うこと。

必要があります。

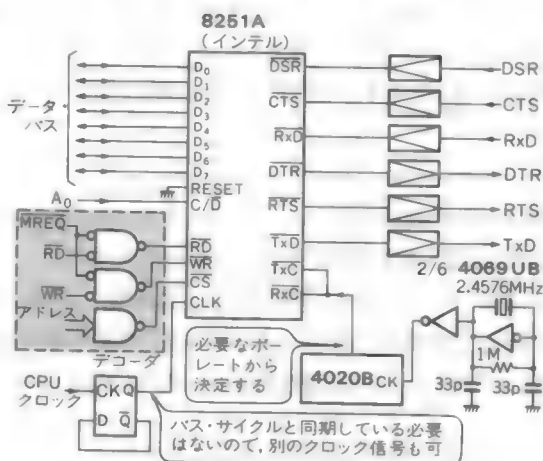
図 5.3 に示すのは直列通信回線用の IC (インテル, 8251A)⁽⁴⁾ を使用したものです。8251A は同期・非同期いずれのモードにも使用可能で、比較的広く使われています。なお、この手の IC としては UART (ウェスタン・デジタル社, 1602 タイプ) や MC6850/52 (モトローラ社), Z80 SIO や DART (サイログ社) などが使われていますが、LAN が広まるにつれ、高機能の IC が続々と登場しています。

なお、8251A の $\overline{\text{RTS}}$, $\overline{\text{DTR}}$ はたんなるビット単位の出力線, $\overline{\text{DSR}}$ はビット単位の入力線ですから、名称のとおり使用する必要はありません。

▶ **電流ループの例** つぎに、通信回線とインターフェースについて述べておきます。通信回線としては古くは電流ループが使われていました (図 5.4 (a), (b))。電流ループは古くさいように思われるかもしれませんが、低速 (1200bps 以下) での長距離通信には、雑音に強く費用が安いので、今でも使われています。同図(a)はよく使われる 4 線のものです。長距離を低費用で引っ張るのなら、同図(b)のように両端局に電源を設け、全部で 2 本の電線で引くのが望ましいといえます⁽¹⁾。

▶ **RS-232C の例** 図 5.4 (c), (d) は RS-232C の例です。(c) は専用の IC を用いたもの、(d)

図 5.3 専用 IC を用いた直列通信回線

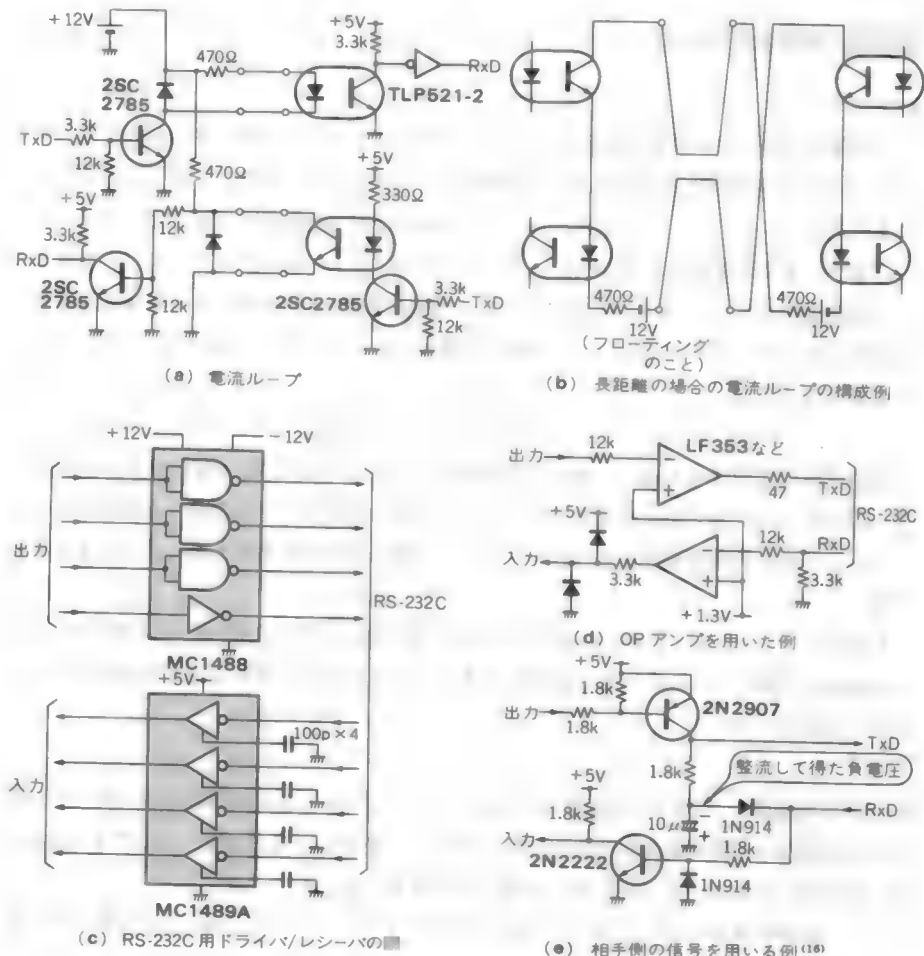


はOPアンプを用いたものです。RS-232Cは $3\text{ k}\Omega$ の受端インピーダンスに $\pm 3\text{ V}$ の電圧があればよく、短距離ならば(d)のOPアンプでも使用可能です。

さらに手を抜いたのが同図(e)で、これは相手側から送られてきた信号を整流して負側の電源としています。これは送り出し側の電圧が $\pm 9\text{ V}$ 以上で距離 5 m 以内ならば使えないことはありませんが、本格的な使用には耐えませんので念のため。

RS-232Cの後継仕様としてRS-422/423が使われようとしています。これらについては

図 5.4 通信回線用インターフェースの例



専用のドライバやレシーバ IC を使うのが無難です (第3章, 第4章を参照)。

なお, RS-232C などの規格では電氣的仕様のほか, コネクタの機械的仕様も細かく決まっています。使用される前に, 参考文献などでよく調べる (たとえば『トランジスタ技術』, 1983 年 12 月号を参照) が必要です。

5.2 映像信号のインターフェース

例 5-2 映像信号の出力

映像信号は 2 次元の情報を直列化して 1 本の信号で送ることができ, モニタ・テレビ (CRT) に映像化すれば視覚に直接訴えられるため, 非常に広く用いられています。市販のパーソナル・コンピュータでも, 標準的な出力装置といえば CRT でしょう。

また, 文字多重放送をはじめとして, いわゆる“ニュー・メディア”の中心となるのも映像信号です。しかし, マイコンにとっては 10MHz もの帯域をもつ映像信号を直接に入出力するのは大変で, 多くの場合, DMA を含めてかなりの量のハードウェアが必要となります。

映像信号を出力するには, 一般には VRAM (ビデオ RAM) と呼ばれる方法を使います。これは, CPU から見ると通常のメモリとして読み書きでき, 映像信号を得るにはリフレッシュ・アドレス発生器からの信号によって一種の局所的な DMA を行おうとするものです。

以前はリフレッシュ・アドレス発生のために, 数十個の TTL を使ったものですが, 最近では高性能の CRT コントローラ (CRTC) が多く出回っており, 特殊な用途を除けばそれを使うのが簡単です。

▶ **CRTC の種類** CRTC の種類は非常に多いのですが, メモリの制御方法に関しては図 5.5 の 2 種類に大別できます。(a) は CPU のアドレス空間にメモリを置くものです。表示内容 (VRAM の内容) は, CPU から直接に読み書きできます。

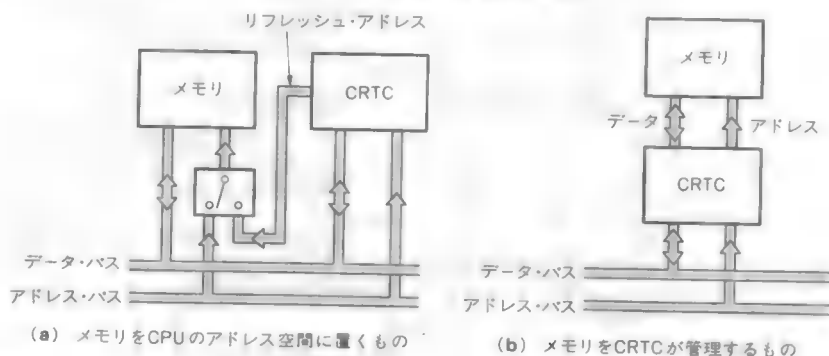
一方, (b) は高性能の CRTC に多く見られるタイプで, メモリと CPU とが分離されてお

り、CPUからのメモリの読み書きはCRTCを介して間接的にを行います。そのかわり、線分や円弧を引く仕事はCRTC側で面倒を見てくれます。

▶文字表示用とフル・グラフィック用 文字表示用でもフル・グラフィック用でも回路の構成は本質的には同じ（文字表示用では文字発生器（キャラクタ・ジェネレータ）が付く程度）ですが、メモリ量に差があります。

すなわち、文字表示用としては標準規格（後述）のCRTでは1画面あたり英数字2000文字程度が限界ですので、メモリも1画面あたり2KBあれば済みます。ところが、フル・グラフィック用では最低でも■～16KB、高度なものでは256KB～1MBのメモリを必要とします。普通の8～16ビットのCPUにとって、これだけのメモリをアドレス空間に置くのは負担が重いので、フル・グラフィック用としては図5.5(b)の構成が多く使われます。

図5.5 CRTCとメモリとの構成



■5.1 映像信号のタイミング

水平周波数	15.75kHz
垂直周波数	60Hz
走査線数	525本*

* 飛越し走査を行った場合で、1フィールドあたり262.5本となる。飛越し走査をしない場合262本でもよい。

(a) NTSC規格

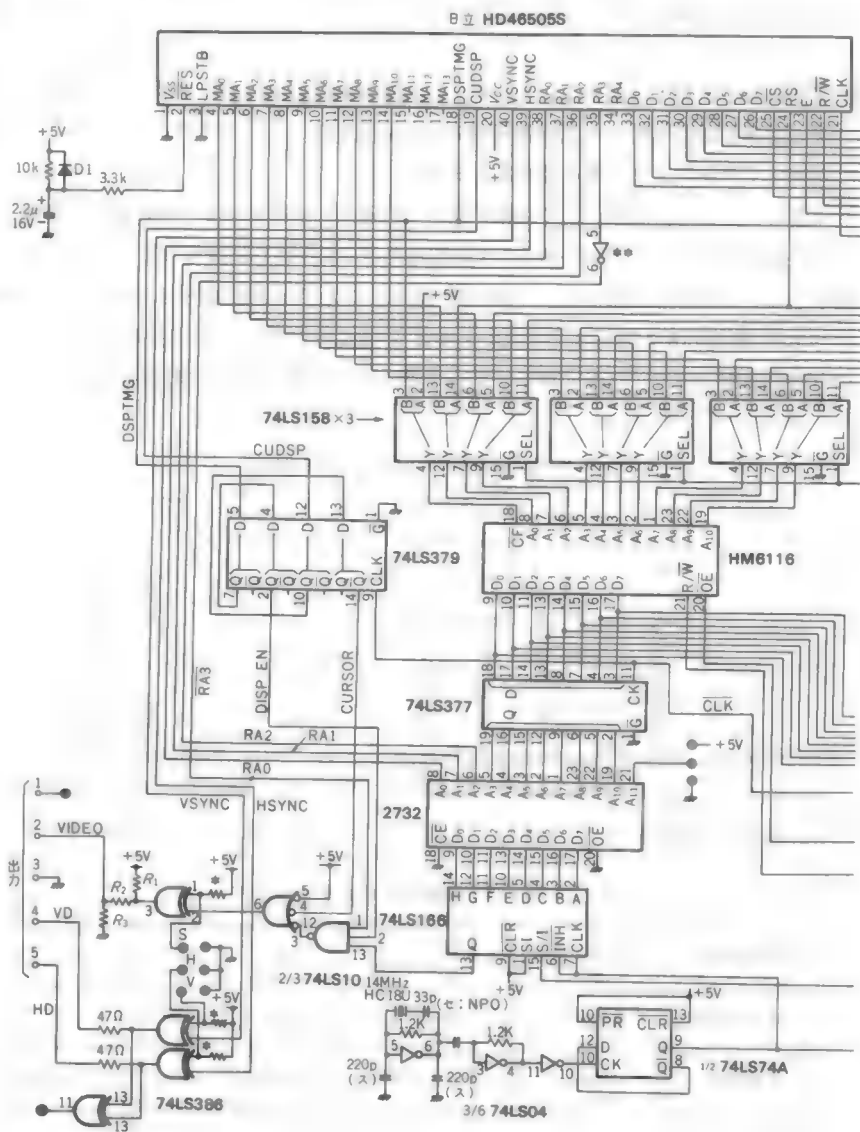
1行あたりの表示桁数	80	64	40
水平総文字数	130	109	65
ドット周波数	14.318MHz	12MHz	$\frac{1}{2} \times 14.318\text{MHz}$

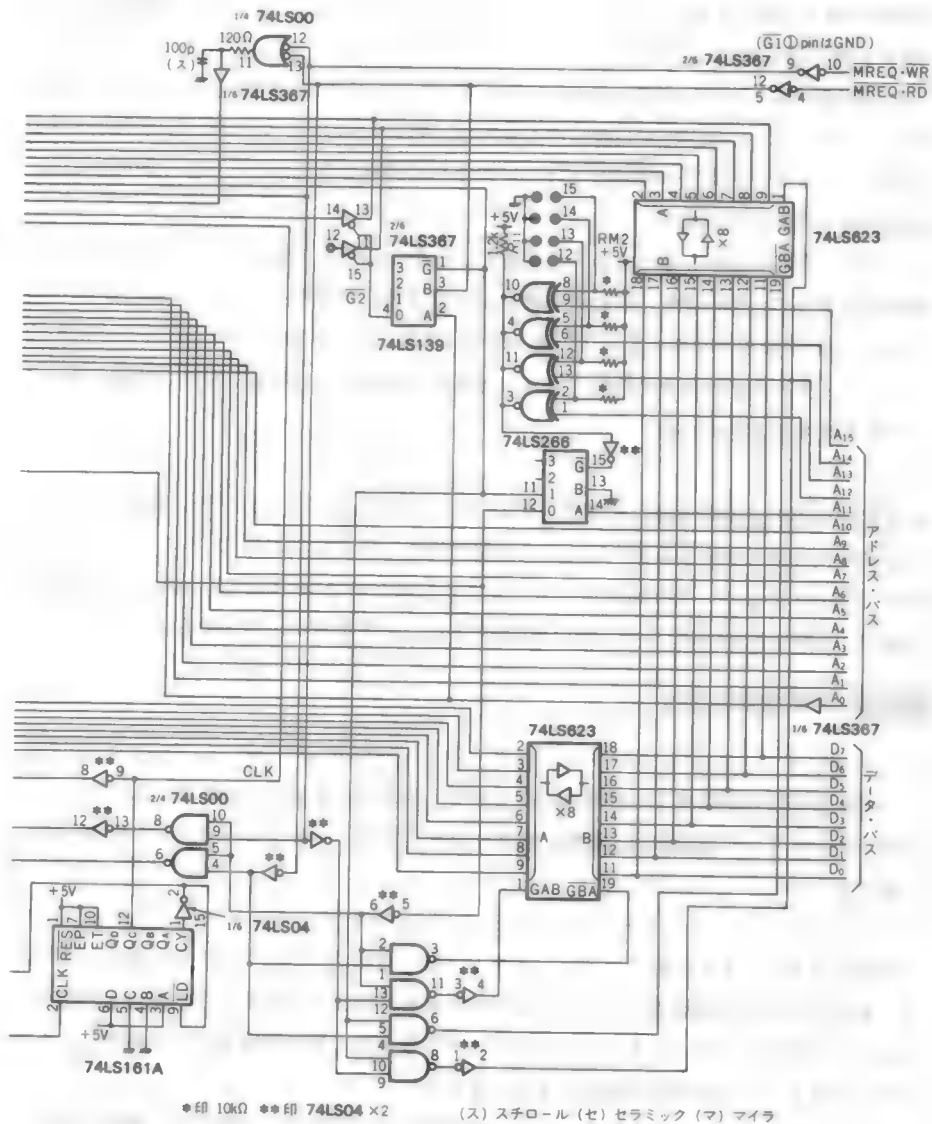
1文字の構成…5×7ドット、字間2ドット
垂直周波数…60Hz

走査線数…262本（飛越し走査なし）

(b) 文字表示用のタイミング例

図5.6 映像信号の出力





▶映像信号の規格 市販のパーソナル・コンピュータでは、高解像度用と銘打って特殊な規格の専用 CRT を使うものがありますが、一般には RS170 など、既存の規格に合った CRT を使うほうが良いでしょう。

RS170 のタイミングは白黒かカラーか (NTSC)、などにより細かく定まっていますが、単にモニタ・テレビ (CRT) に映すためならば、■ 5.1 (a) (155ページ参照) を目安とすれば良いでしょう。明るさの階調をつけないのならば飛越し走査はしないほうが見やすく、走査線数は 262 本とします。

とくに、文字表示用を考えるとこの規格と 1 行あたりの文字数とから各タイミングは一意的に決まってしまいます。その例を表 5.1 (b) に示すので参考として下さい。

なお、CRT は多少のタイミング誤差 (数%) があっても平気ですが、VTR (ビデオ・テープレコーダ) などのビデオ機器ではもっと厳密に規格通りの信号でないと作動しないことがありますので念のため。

▶文字表示用の映像信号出力インターフェース 図 5.6 (156, 157ページ参照) に示すのは産業用の文字表示用インターフェース基板の例で、日立の CRT コントローラ HD46505 S⁽¹¹⁾ を用いています。この IC は E 入力の立上りに対して R/\overline{W} 信号にセットアップ時間が必要で、Z80 など 68 系以外の CPU に接続するには E 信号を少し遅らせる必要があります。

例 5-3 映像信号の入力

映像信号は非常に多くの情報を含んでおり、図形などを取り込むには有用です。しかしサンプリング周波数が 10MHz 程度になりますから、かなりハード的な処理が必要です。

映像信号を出力する装置としては、テレビ・カメラのほかに、VTR やビデオ・ディスク、あるいは放送受信用のチューナなどがあります。これらのなかには外部から同期をかけることが可能なものと、そうでないものがあります。外部同期をかける場合と、そうでない場合とでは、処理の方法が若干異なります。

図 5.7 に示すのは、外部同期をかけた工業用 TV カメラの出力を用いて、画面に映った白い物体の面積を求める回路です。

このような回路では同期信号発生用の IC を使うのが簡単です。しかし、使い方や入手

が容易なものは多くありません。図 5.7 で使ったソニーの CX2930A は、14.31818MHz ($3.579545\text{MHz} \times 4$) の水晶発振子を接続すれば NTSC (PAL, PALM, SECAM も可) 規格の同期信号を出力します。電源も +5V で使いやすいのですが、パッケージがミニフラット型で試作に不便なのが欠点です。

画面の 1 ドット (画素) の縦横比を 1 : 1 にするには、走査線数 262.5 本に対してドット・クロック周波数を約 6MHz とします。ここでは、74LS132 によるヒステリシス発振器

図 5.7 面積測定回路

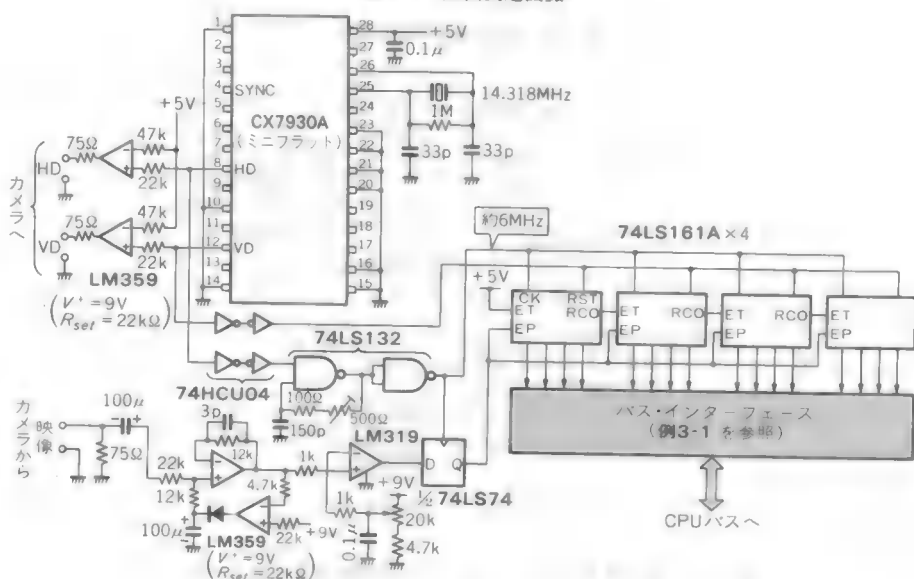
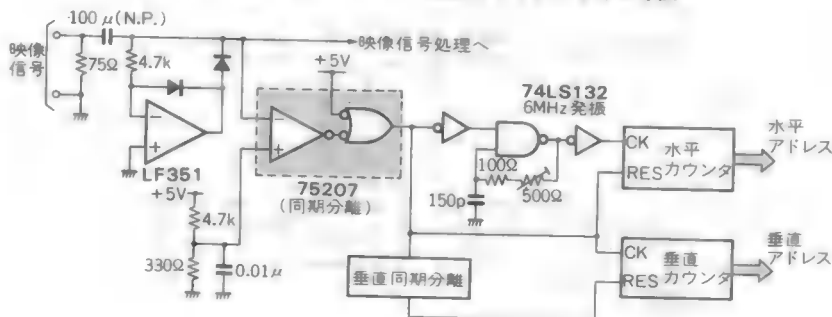


図 5.8 映像信号から水平・垂直のアドレスを作る方法

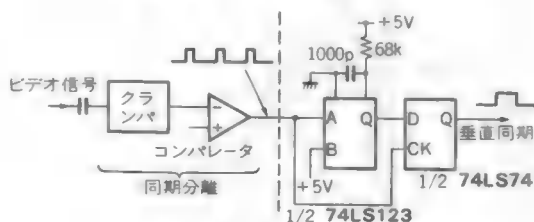


を使っていますが、電源電圧や温度が変化すると、周波数が最大で数%変動します。もし高い精度を必要とするならば、PLL を使って 14.318MHz と同期させるのがよいでしょう。

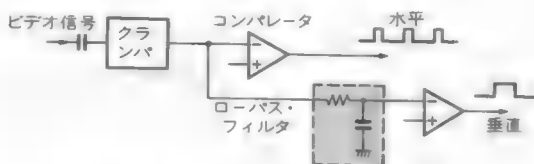
図 5.8 に示すのは、外部同期をかけられない装置からの映像信号の入力方法です。この場合は、映像信号から同期分離を行い、そのなかから垂直同期信号を抽出し、場合によっては飛越し走査の各フィールドを区別しなければなりません。

図 5.9 (a) に示した垂直同期検出回路はデジタル的なもので、同図(b)のローパス・フィ

図 5.9 垂直同期の検出方法

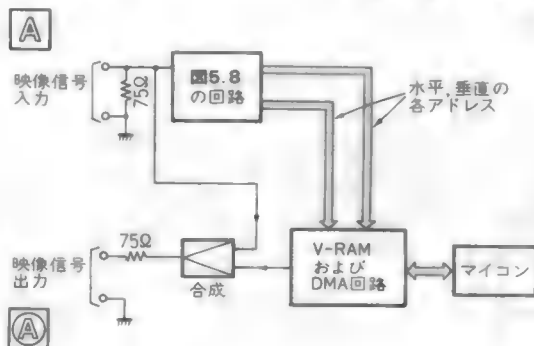


(a) デジタル的な方法



(b) ローパス・フィルタによる方法

図 5.10 他の映像信号とマイコンからの映像とを重ね合わせる



ルタ型のものに比べて簡単で確実です。水平のドット・クロックを与えられた映像信号に追従させるには、PLL が必要となります。

図 5.8 の回路を用いて図 5.10 のような構成とすれば、外部の映像信号とマイコンで作った画面とを重ねることができます。画面を重ねる機能を含んだ CRT コントローラ用 IC もありますが、図 5.10 の構成とすればドット数などが好きなように設計できます。

5.3 分散処理のインターフェース

例 5-4 分散処理 (複数 CPU システム)

マイコンにとってもっとも苦手な高速かつ並列的な処理を行うには、複数 CPU システムを考慮します。CPU の価格が非常に安くなった今日、仕様 (部品数など) が許すのなら、仕事を細かく分割したほうがシステム構成はむしろ単純です。

ここでは、簡単な例として共通バスをもたない複数 CPU システムの CPU 間のやりとりについて説明します。

たとえば、DMA を用いて映像信号を出力する場合、実効的な処理速度は半分程度に低下します。とくに、外部からの割込みなどに対する応答速度が悪化し、ただでさえ不足がちなマイコンの (見かけの) 速度を一層低下させています。これを防ぐ手段はいくつかありますが、ここでは複数 CPU システムを考えます。

図 5.11 に示すのは分散処理を大幅に取り入れたシステムの概念図です。これは複数 CPU システムとしては非常に簡単な構成で、とくにデータの流れが単純なため、ハードウェアの設計も容易です。

図 5.12 に示すのは実際の例で、入力処理部は外部クロックに応じ、 $20\mu\text{s}$ 以内に外部からのアナログ信号を取り込む必要があります。マイコン (ホスト) 側はそのデータを処理するとともに、画面にデータを表示します。

低価格帯の市販パーソナル・コンピュータでは、DMA オーバヘッドが加わり、 $20\mu\text{s}$ の応答時間は実現できません。また、ハードウェアだけで構成するのも結構大変です。そこ

* 画面用のバッファ・メモリを設け、実質的な DMA オーバヘッドを下げることが多い。

でこの例では、スレーブCPUはデータの入力だけを受け持ちます。

2個のCPUのデータ転送は、同一のメモリに対して両方のCPUから同時に読み書きを行う可能性がなければ比較的簡単です。

図5.13(a)に示すのはもっとも簡単なもので、一方からは出力ポート、他方からは入力ポートと見なせます。双方向のやりとりが必要な場合、これを2組設けます。この回路では、

図5.11 分散処理化を図ったシステムの例(各CPUのバスは共通でないもの)

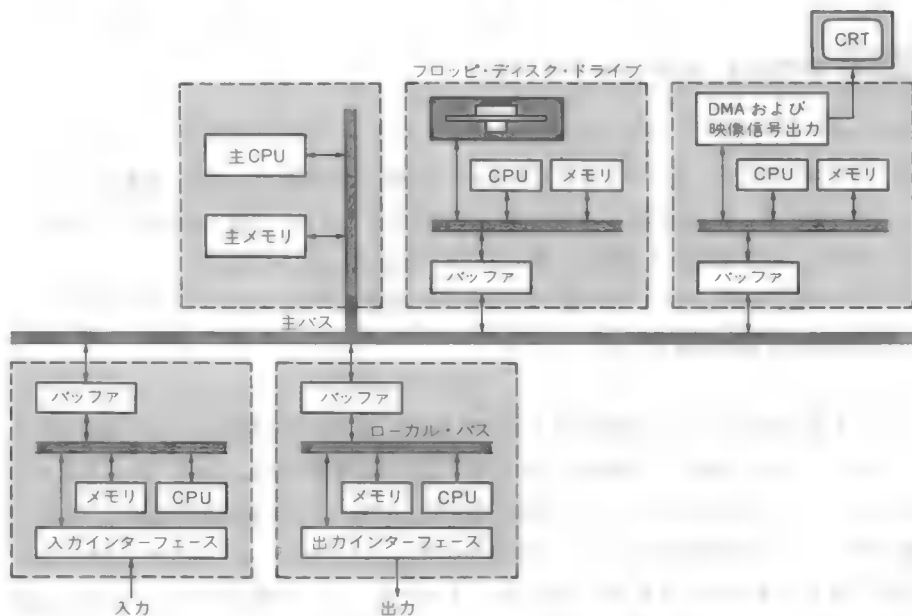


図5.12 複数のCPUを使った簡単なアナログ入力インターフェース

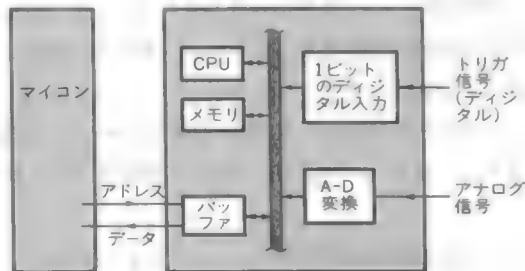
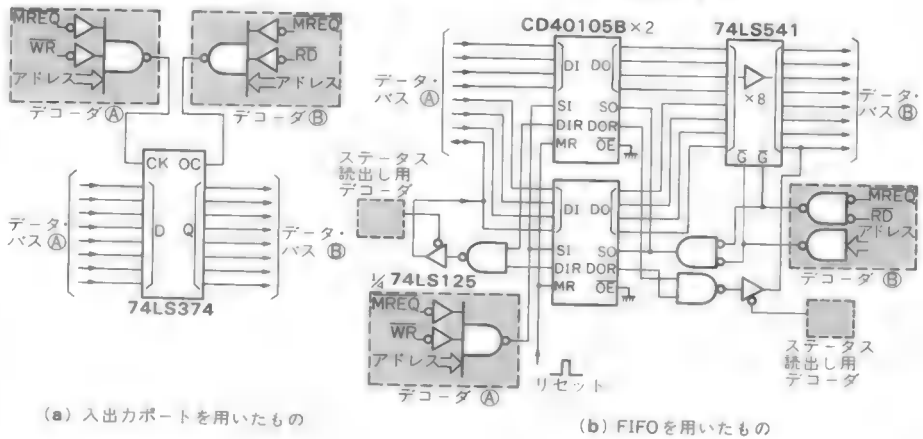


図 5.13 複数 CPU 間のデータのやりとり (一方向のもの)



(a) 入出力ポートを用いたもの

(b) FIFOを用いたもの

書き込み中に他方からデータを読み出した場合、データが保証されませんから注意が必要です。

同図(b)に示すのは FIFO⁽⁹⁾ を使ったものです。この回路の利点は、読出し側と書き込み側とがまったく非同期的に作動してもデータが失われないことです。ただし、FIFO が満杯になればダメなので念のため。

より複雑な分散処理を行う場合、バスを共有することになります。この場合は各 CPU のバス使用タイミングを調停する手段が必要となります。本書ではこれには触れませんが、バス調停用の IC も作られており、今後はそのようなシステムも増えると思います。

第 6 章

入出力インターフェース・ユニットの実際

第3章～第5章では個々の入出力インターフェースの設計法を説明しました。実際のシステムでは、以上の方法をもとにして必要とされる入出力インターフェースを設計すれば良いのですが、似たようなインターフェース回路を毎回設計するのも大変です。そこでメーカーの多くは各種の標準ユニットを用意しており、必要に応じてそのユニットを組み合わせ、システムが構成できるようにしています。

これらのユニットの中には、

- (1) 特定のパーソナル・コンピュータの拡張スロットに差し込むためのもの。
- (2) 業界標準規格に従っていて、他社のものとも任意の組合せが可能なもの。
- (3) 同一シリーズの基板との組合せしか考慮していないもの。

などがあり、基板やコネクタの形状、信号線の電氣的な規格などが、各々に定められています。

ここで紹介するのは上記(3)に含まれるものです。このシリーズの基板の特徴としては次のような項目があげられます。

- (1) Z80 CPU を使い、実績のある（いわゆる“枯れた”）ハードウェア/ソフトウェアを利用できる。
- (2) ソフトウェアの負担をなるべく少なくするよう考慮してある。
- (3) 入出力の多くはメモリ・マップとなっており、割込みは通常は使わない。

これらの点から、特定のパーソナル・コンピュータ用の入出力モジュールなどとは設計方針が異なっていますが、現場での入出力インターフェースとして参考になる部分が多いでしょう。

表 6.1 ユニット基板の1例

分 類	内 容
CPU基板	CPU, メモリ, デバッグに必要な入出力など
入出力基板	絶縁されたデジタル入出力(各16本)
	アナログ入力(8ビット×8チャンネル)
	アナログ出力(12ビット×4チャンネル)+デジタル入出力
	スイッチ読み込み用(16回路×16接点)
	■操作作用(スイッチ, LED, ポテンシオメータなど)
	その他数種類(表示用など)
メモリ基板	28p スタティック・メモリ×8個 (ROM/RAM兼用, 最大64K バイト)
スレーブCPU基板	直流モータ制御用の入出力を持つスレーブZ80
その他	映像信号の取込み用, VRAMなど, 数種類

●全体の構成

このシリーズに含まれる基板の1例を■6.1に示します。一般に、ロボット制御などのシーケンサ的な応用では、リレーやスイッチ類、あるいはオープン・コレクタの入出力が大半を占めます。それ以外では直流サーボ用のアナログ出力と、表示関係(LED、液晶、またはCRT モニタ)出力とがあれば、ほとんどの場合に間に合います。

一方、プロセス制御に使う場合はアナログ入力が必要となります。これは低速・高分解能(12~14ビット、毎秒10回程度)と、高速・低分解能(8~10ビット、毎秒10,000回程度)との2種類が必要です。

また、最近は工業用テレビ・カメラ(ITV カメラ)やイメージ・センサを使う機会も多く、そのためのインターフェースも考慮してあります。

このようにユニットを組み合わせて色々なシステムに応用しようとする場合、何があれば必要かつ十分であるか、良く検討することが重要です。

6.1 CPU 基板

CPU 基板(付図6.1に全体の回路図を示す)はすべての装置に少なくとも1枚は使用され、大半の用途では“心臓部”となります。したがって、CPU 基板の設計は、このようなユニット基板のシリーズを構成する際、非常に重要となってきます。

■ CPU 基板の特徴

CPU 基板の中心はいうまでもなく CPU です。このほか、一般にはクロック発振回路、リセット回路、バス・バッファなどが必要となります。

本来、このようなユニットのシリーズでは、メモリや入出力インターフェースを別々の基板とし、用途に応じてそれを組み合わせるような使われ方をします。したがって、CPU 基板にはメモリや入出力インターフェースは載せなくても構いません。

しかし、どのような用途でも多少のメモリは必要（RAM なしのシステムはあり得るが、ROM なしは一般にはあり得ない）ですし、最低限の標準的な入出力インターフェースはデバッグや小規模の用途に便利です。このため、この基板にはメモリと若干の入出力インターフェースが設けられています。ただし、これらはデバッグや割込みなど特殊な用途に使われることを主眼としており、他のインターフェース基板とは設計が若干異なる点に注意して下さい。

● CPU の選定

このシリーズは、シーケンス・コントローラ的な用途、あるいはプロセス制御用などを主眼としています。これらの用途では処理速度はさほど重要ではありません。むしろ、簡単かつ確実なハード/ソフトで動作することが要求されます。

この基板の設計時点（1982 年）ではハード/ソフトともに使用実績があり、かつ開発環境の整った CPU としては Z80、8085、6802 などの汎用 8 ビット機しかありませんでしたので、その中から Z80 を使用しました。その後、Z80 は（その高速版 Z80A、Z80B を含む）各社で量産され、安定かつ安価に供給されました。

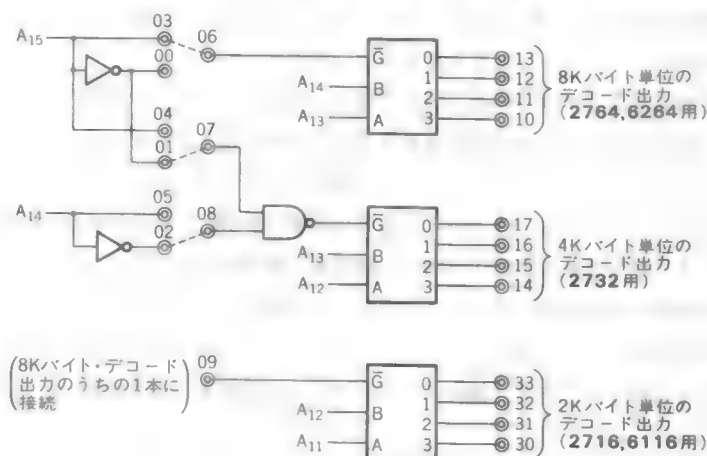
しかし、これから CPU 基板を設計するのなら、シングルチップ・コンピュータ（8048/51、Z8、6301 など）を使って部品数を減らすか、より高機能の CPU（8088、6809 など）を使って性能を稼ぐかを選ぶ必要があるでしょう。

● メモリ

CPU 基板にメモリを載せるか否かは意見の分かれるところです。しかし、ROM や RAM が各々 2 K バイトずつもあれば充分かつ簡単な用途は少なくありません（むしろ、大半の用途がその程度だといえる）。そのような場合に CPU 基板単体でも用がすむよう、この基板には 28 ピンのスタティック・メモリ用のソケットが 4 個用重してあります。

設計時点では ROM は 2716（2 K バイト）や 2732（4 K バイト）が、RAM は 6116（2

図 6.1 アドレス・デコーダの構成



K バイト) が全盛で、この基板に実装できるメモリも $2716 \times 2 + 6116 \times 1$ (ソケット1個は予備) の合計6K バイトといったところでした。しかし、メモリの進歩は早く、1984年の段階ではROM, RAMともに8Kバイト型(2764, 6264)が主流となり、CPU基板上に24Kバイト(1個は予備)が実装できるようになりました。これはかなりの量で、ほとんどの用途でメモリ基板は不用となっています。

なお、使用するメモリ(ROM 2716/2732/2764, RAM 6116/6264の各型)およびそのアドレスは、ジャンパ(ショート・ピン)で選択できるようになっています(図6.1)。

●直列通信回線(RS-232C インターフェース)

この基板には8251Aを中心としたRS-232Cインターフェースが設けられています(図6.2)。実際の応用では直列通信回路を使わない場合もありますが、デバッグの際、このような標準的なインターフェースがあると標準のモニタ・プログラムが使用でき、非常に便利です(ハード/ソフトのデバッグの手順については巻末も参照して下さい)。

図からおわりのように、この基板では8251AのクロックとCPUのそれとが共通になっています。これは、設計が古くクロックとして2.5MHz(ボーレートの関係から2.4576MHzを使うことが多いが)を主眼としていたためです。CPUにZ80Aを使い、クロックを4MHzにするには、2分周した2MHzのクロックを8251A(およびZ80A CTC)に与えるのが適当です(図6.3を参照)。

RS-232C 用のレベル変換にはポピュラーな 1488/1489A を使用してあります。また、ドライバ側に必要な $\pm 12V$ の電源はカードエッジ・コネクタを通して外部から供給します。

● 並列インターフェース

意外に思われるかもしれませんが、この Z80 PIO を使った並列インターフェースはほとんど使われることがありません。しいて使われる場合を掲げると、基板上の DIP スイッチの状態を読む必要がある場合と、Z80 PIO を通して割込みをかける場合くらいなものでしょう。この部分があまり使われないのは、次のような理由からです。

- (1) Z80 PIO の出力は、そのままでは長いケーブルを駆動できない。
- (2) 必ず初期設定を必要とする。これは手間がかかる以外に安全性（入出力を間違えたりした場合に危険）の面で問題がある。
- (3) シーケンス・コントローラやプロセス・コントローラなどの用途では多数のデジタル入出力を必要とする。このためデジタル入出力専用の基板（後述）を組み合わせる使用なので、CPU 基板で入出力を行う必要がない。

したがって、この部分は全く省略してもとくに差し支えはありません。ただし、このシリーズでは CPU 基板以外の基板は原則としてメモリ・マップトになっており、Z80 CPU の割込み機能をフルに使うには不便です。そのため、この Z80 PIO を割込みコントローラ的に使用すると便利ことがあります。

● その他

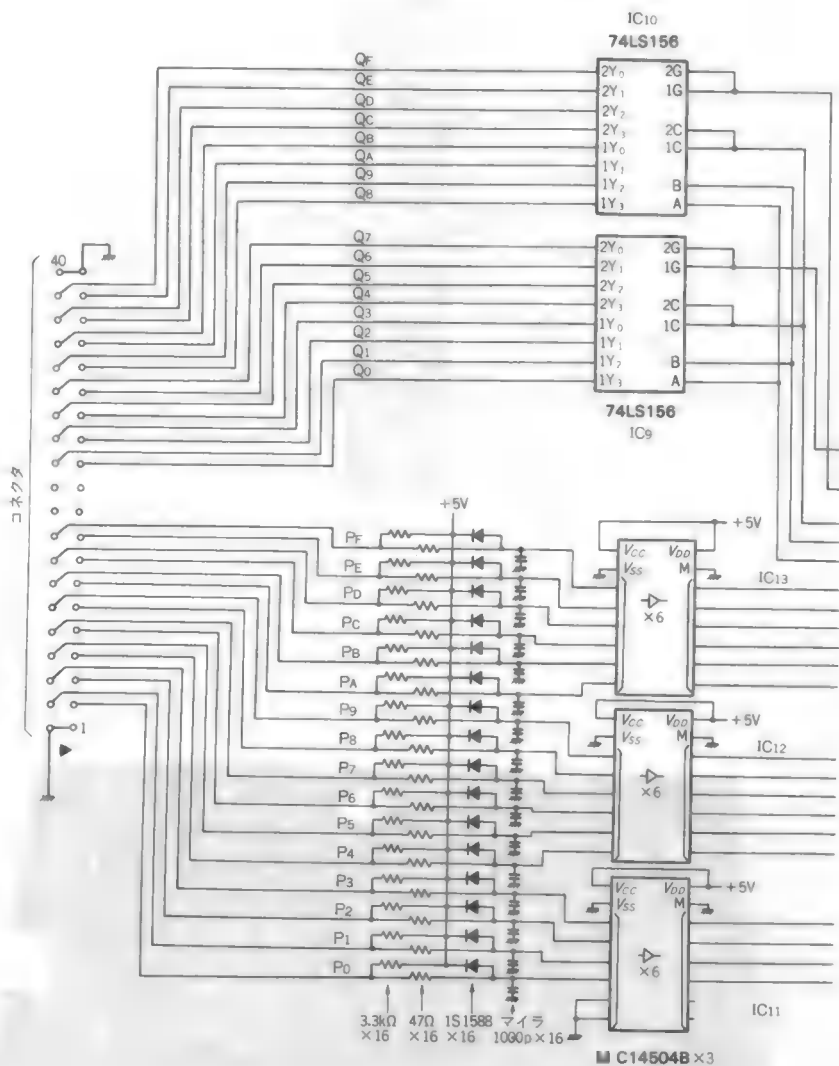
マザー・ボードに差し込み他の基板と組み合わせて使うため、データ・バスおよびアドレス・バスはバッファを介してカードエッジ・コネクタに接続されています（図 6.4）。

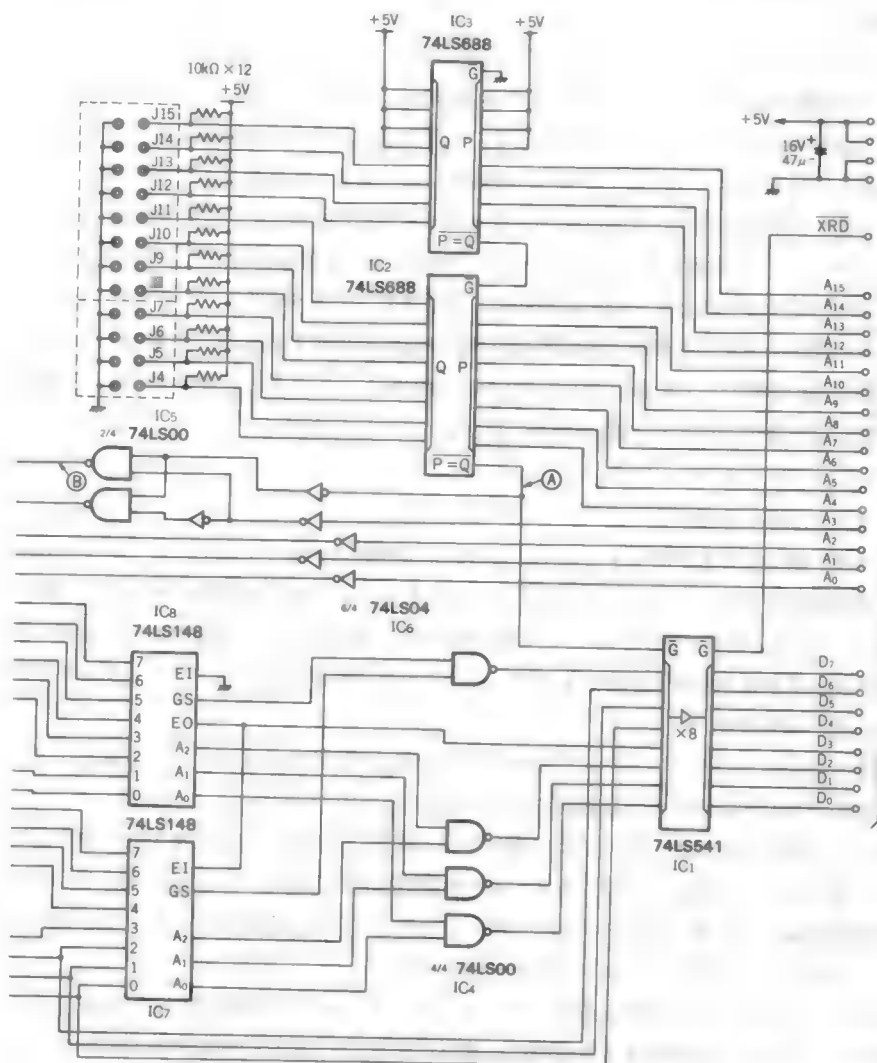
前述のように、このシリーズは標準規格等に合わせたものではなく、独自の接続となっています。なお、基板寸法は $114.3\text{mm} \times 203.2\text{mm}$ です（写真 6.1 参照）。

6.2 多数のスイッチの入力インターフェース

制御装置などへの応用では、パネルに取り付けられた多数のスイッチを読み込まなければならないことがあります。一口にスイッチといっても、トグル・スイッチ、押しボタン・スイッチ、ロータリ・スイッチ、デジタル（サム・ホイール）スイッチなど多くの種類があります。しかし中でもロータリ・スイッチは 1 個のスイッチで多数の接点（1 回転あ

図 6.5 スイッチ入力インターフェース





たり 12~24 接点)を持ち、パネル面積がかぎられている場合にセレクトとして使用するには最適です。

ところが、これを読み込む側は大変で、16 接点のスイッチが 16 個も並べば、それだけで 272 本の配線が必要となります(各 1 本の共通点をお忘れなく)。また、これを 256 ビットとして読み込んでソフトウェアで解読していたのでは、ソフトの負担も大きくなります。

図 6.5 に示すのは、このような場合に使用するスイッチ入力インターフェースです。しかしパネルに取り付けられるのはロータリ・スイッチだけではないので、個別のスイッチ(トグル・スイッチや押しボタン・スイッチ)、あるいは 2 進符号出力のデジタル(サム・ホイール)スイッチなども一緒に読み込めるよう考慮してあります。

一方、長いケーブルを使って電源系統の異なる機器に接続する場合とは違い、パネルまでは数十 cm 程度でとくに強力な雑音源も近くに存在しないと考えられますから、一般的な雑音対策を施すだけとし、絶縁等は考慮しないことにします。

●ダイナミック・スキャン

第 3 章、例 3-7 でも説明しましたが、このような場合はダイナミック・スキャンを利用する方法があります。しかし、ダイナミック・スキャンにも様々な種類があります。図 6.6(a)に示すのは、まずソフトウェアで行データを R_i に書き込み、次に読み出した列データからスイッチの接点位置をさがすもので、ハードウェアは簡単ですが、ソフトウェアの負担は大きくなります。

図 6.6(b)に示すのはソフトウェアでの行データ書き込みが不要のものです。つまり、読出し動作のアドレスに応じて、自動的に 1 本の行をアクティブにします。

さらに、図 6.6(c)に示すものでは、読み出した列データのエンコードまでハードウェアで行います。多くの場合、多接点のスイッチの接点位置は“数値”として扱うか、“テーブルを参照する”のに使うかのいずれかです。したがって、接点位置はエンコードして数値としておいたほうがソフトウェアの負担が減ります。

ただし、特定のビット(接点位置)が“1”か“0”かだけを調べたいこともあり、また同じハードウェアを個別スイッチ用にも兼用できるよう、図 6.5 の回路では列データ(16 本)をエンコードした信号(4 ビット)、スイッチが接続されているか否かを示す信号(1 ビット)、列データの一部のビット・パターンをそのまま出力する信号(3 ビット)を CPU から読み出せます。これで、16 接点のロータリ・スイッチ 16 個(配線引出し本数 32 本)、または個別スイッチ 48 個(同 19 本)、あるいは各種スイッチの混合使用が可能となってい

図 6.6 ダイナミック・スキャンの方法

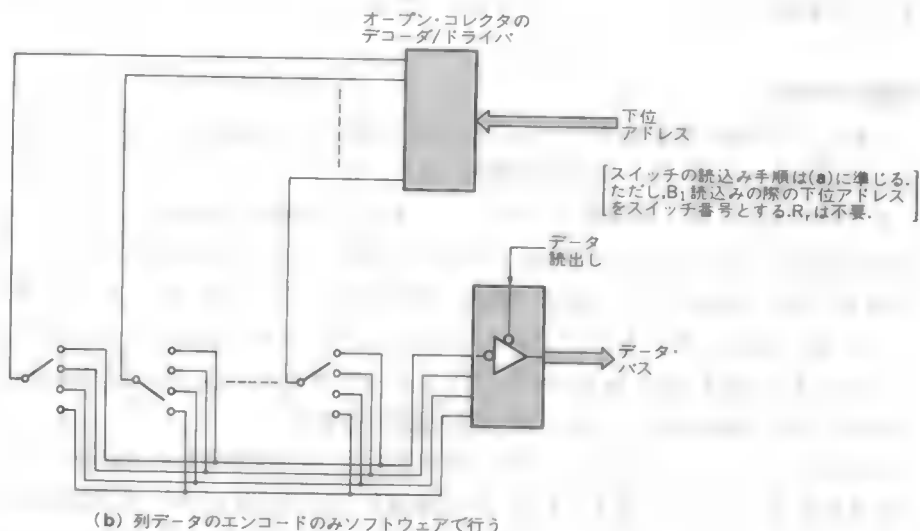
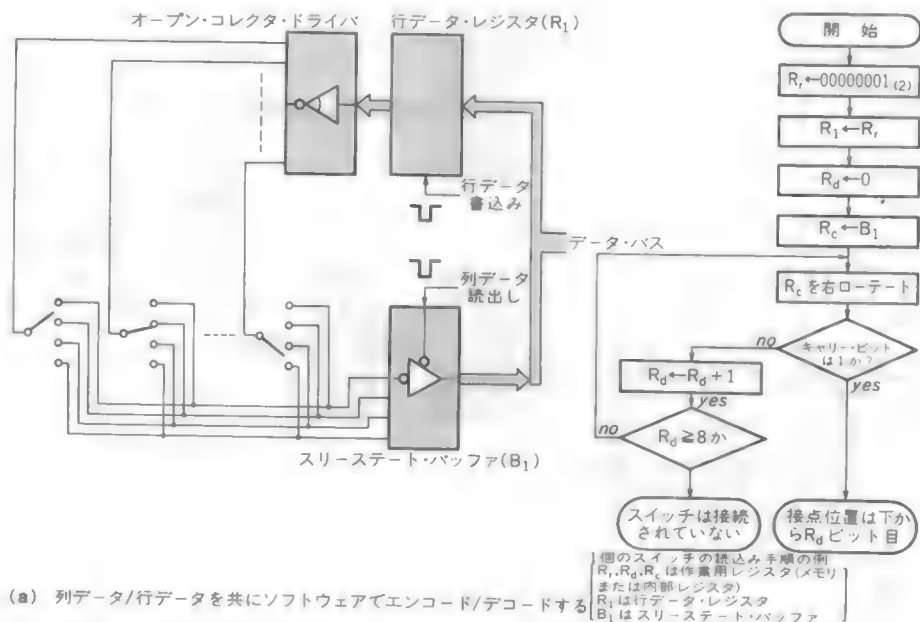
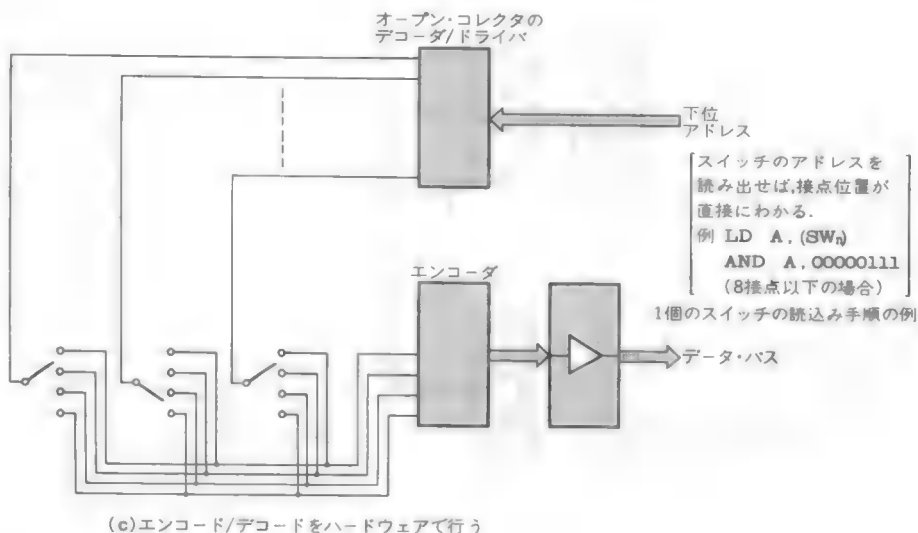


図 6.6 (つづき)



ます。

このように、ハード/ソフトの分担と、ハードウェアの汎用性とを良く考えて仕様を定めることが重要です。

●動作の

つぎに、この回路(図 6.5)がアドレス FFF0 H~FFFFH に設定されている(J4~J15 がすべて開放)と仮定して、その動作を説明しましょう。

まず CPU が LD A, (FFF0) を実行したとします。この命令の最後のバス・サイクルでは CPU からアドレス・バスに FFF0 H が出力されます。このうち上位 12 ビット A₁₅~A₄ は IC₂~IC₃ で比較され、一致出力(④点)がアクティブ“L”となります。また、A₃ が“0”となって⑧点が“L”となり、A₂~A₀ が“0”であることから Q₀ のラインがアクティブ“L”となります(A₀~A₂ にバッファとしてインバータが入っていることに注意)。これにより、Q₀ に接続されているスイッチだけが選択されます。

このスイッチのポジションが“5”であったとすると、P₀~P_F のうち P₅ だけが“L”になります。IC₁₁~IC₁₃ でこれを TTL レベルに変換し、IC₇~IC₉ のエンコーダに印加します。エンコーダはこの結果を 2 進変換し、バス・インターフェース(IC₁)に加えます。少

し遅れて $\overline{\text{MEM}} \cdot \text{RD}$ 信号がアクティブになり、 IC_7 からデータ・バスの下位 4 ビットにスイッチの接点位置 “5” が出力されます。

このとき、 $D_4 \sim D_6$ には入力 $P_0 \sim P_2$ の状態がエンコードを通さないで接続されていますから、1 個のロータリ・スイッチのかわりに 3 個の個別スイッチを接続できます (図 6.7)。また、 D_7 にはエンコードから接続の有無を示す信号が接続されていて、接点数の拡張にも使えます (図 6.8)。

図 6.7 スwitchの接続

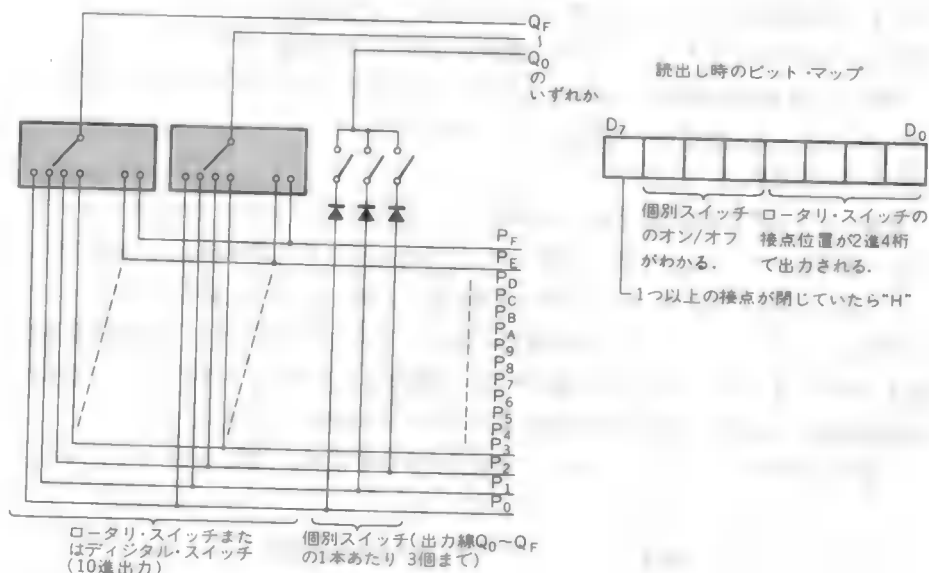
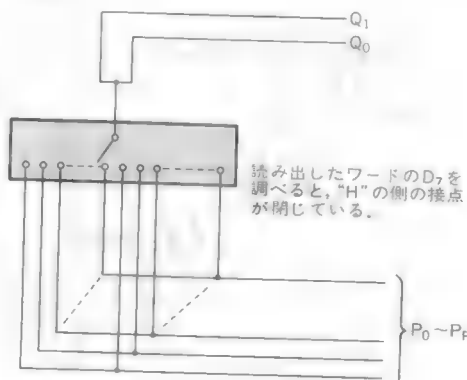


図 6.8 17 接点以上のロータリ・スイッチの接続



6.3 フォト・カプラで絶縁された汎用入出力インターフェース

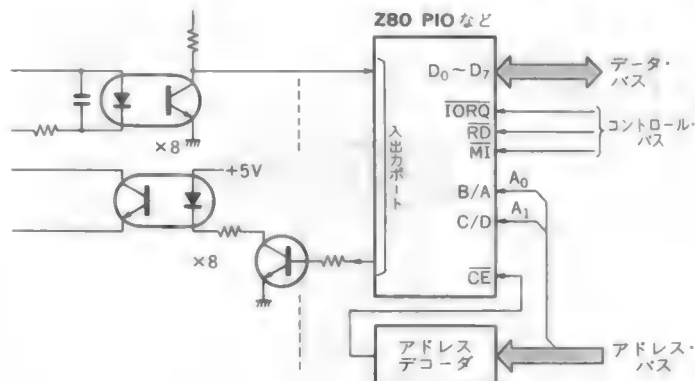
シーケンス・コントローラ的な用途では、多数のデジタル入出力が必要とされます。多くの場合、この信号は接点またはオープン・コレクタの形で出し入れされます。これはレベル変換が容易であるとともに、電源系統の異なる装置間での干渉を防ぐためです。また、一般にモータやソレノイドを使う場所（シーケンス・コントローラが置かれるのは、そのような場所に決まっていると思ったほうが良い）は雑音環境が良くないと考えられますから、入出力はフォト・カプラなどで絶縁しておくほうが無難です。

付図6.2（巻末繰込図を参照）に示すのは、このような目的のために作られた入出力インターフェース基板です。入出力は各16本ずつとなっており、すべてフォト・カプラで絶縁してあります。

通常、このような用途にはZ80 PIO（ザイログ）や8255（インテル）、6821（モトローラ）などの並列インターフェース用ICを使いたくるところですが（図6.9）、この基板ではすべてTTLを使っています。これはCPU基板の項でも述べましたが、並列インターフェース用のLSIは、ソフトウェアでの初期設定が必要で、それを間違えるとICを破損する可能性すらあり、またフォト・カプラを駆動するのに駆動用IC（またはトランジスタ）を用いる必要があり、必ずしも部品実装面積が減らないなどの理由によります。

一般に、並列インターフェース用LSIの長所が生かせるのは、割込みやハンドシェイク

図6.9 並列インターフェース用LSIの利用



機能を活用する場合にかぎられます。

●アドレス・デコーダ

この基板の入出力は各 16 本 (2 バイト) ずつで、出力側のデータの読み返しができるので 4 バイトのアドレスが必要です。用途によっては、ディジタル入出力の本数が 16 本ずつでは足りないこともあり、その場合はこの基板を 2 枚以上使いますが、その場合にもソフトウェアから見て連続したアドレスになったほうが何かと便利です。

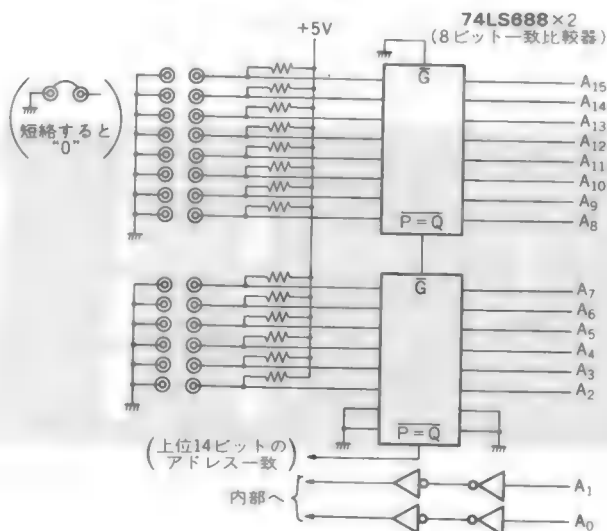
以上の点から、この基板は上位 14 ビットのアドレスのフル・デコードができるようにしてあります。アドレス・デコーダとしては 8 ビットのコンパレータ (74LS688) を使っています (図 6.10)。

●入出力の極性の変更など

この回路では、入出力ともに接点がオンした状態がソフトウェア上での“1”に、オフが“0”に各々対応しています (図 6.11)。これを逆にするには、付図 6.2 の回路中の $IC_1 \sim IC_{12}$ を非反転の 74LS253 に、 IC_1 を同じく 74LS541 に変更します。

また、図の回路は入力用として絶縁された 12V の電源を仮定していますが、用途によっ

図 6.10 アドレス・デコーダ



ては5Vや24Vになることもあります。その際にはコネクタ側の抵抗を変更し、フォト・カプラの1次側に10mA前後の電流が流れるようにします。

●入出力の本数と汎用性

写真6.2からおわかりのように、この基板はガラ空きです。基板寸法と回路規模とを考

図 6.11 ソフトウェアから見た論理(1/0)の対応

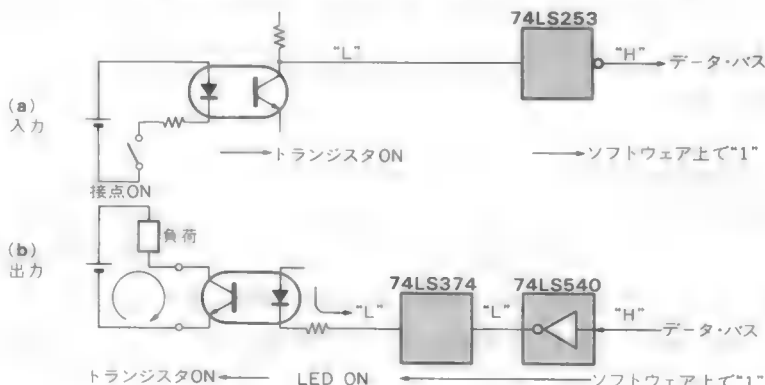


写真 6.2 フォト・カプラで絶縁された入出力用ボード

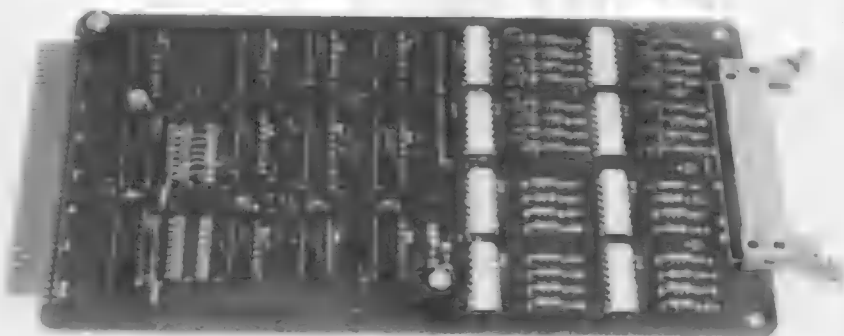
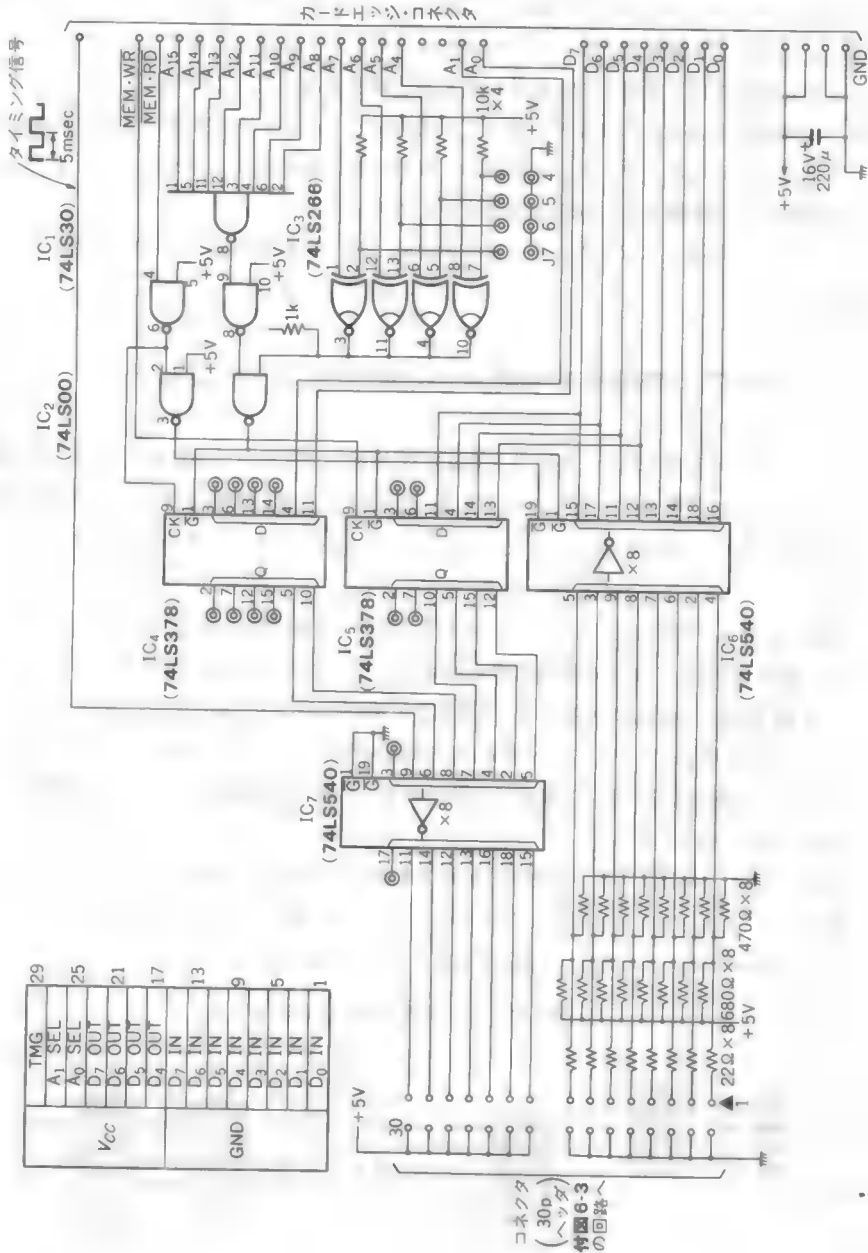


図 6.12 操作部用インターフェース



えると、入出力とも各 32 本（現状の 2 倍）くらいのが収められそうです。そうできれば、この基板を 2 枚必要としていた用途も 1 枚ですみ、何かと便利のように思えます。

しかし実際には、基板からの信号引出しが困難となるので、これくらいが実用的でしょう。この基板では 40p のヘッダを使っていますが、これ以上のピン数のものは基板幅を考えると取り付けられません。ピン間隔がより狭い (1.27mm) コネクタもありますが、価格や基板製作上の問題から、採用は慎重を要します。

このような基板ユニットのシリーズを設計する際、最も問題となるのは上記のような点なのです。

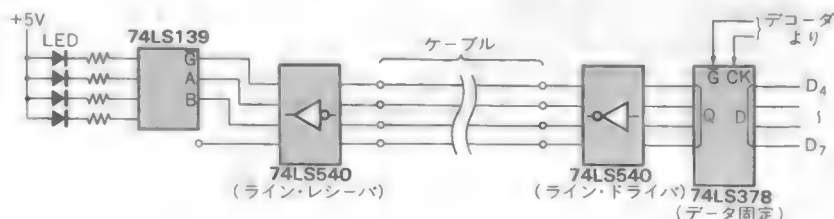
6.4 ケーブルの延長を考慮した入出力インターフェース

スイッチの入力や LED への出力、あるいはポテンショメータの位置の入力などを離れた場所で行いたい場合は少なくありません。その場合、単に配線を延長したのでは配線本数が増加したり、雑音に弱くなったりして好ましくありません。そのような場合は延長を考慮したインターフェースを行います。

図 6.12 および付図 6.3 はスイッチや LED の付いた操作箱用のインターフェースの例です。図 6.12 はシステム本体側に組み込まれ、ここから操作部まで 18 対のシールド付ツイスト線で 10m の延長が可能です。付図 6.3 は操作部の例で、数字などの 20 キーのキーボード、2 桁のデジタル・スイッチおよび 8 個の個別スイッチ、1 個のレバー付ポテンショメータ、4 個の LED ランプが設けられています。この例は搬送用ロボットの操作用として設計されたものです。

まず、LED (4 個のうち 1 個だけ) を点灯するための書込み時には、CPU からのデータ ($D_4 \sim D_7$) は IC₅ でラッチされ、ライン・ドライバを兼ねた IC₇ から付図 6.3 の IC₁₂ を経由して IC₁₇ でデコードされ、LED を点灯します (図 6.13)。

図 6.13 データ出力時の信号の流れ



スイッチやポテンショメータの状態を読み出す場合には、 \overline{RD} の前縁でアドレスの下位2ビット (A_0 , A_1) をラッチし (IC_4), その状態を操作箱側に送ります。この信号は IC_{12} を経てマルチプレクサ ($IC_{13} \sim IC_{16}$) に印加され、必要なデータを選択します。

マルチプレクサで選択されたデータは IC_{11} からライン・レシーバ兼バス・インターフェースの IC_9 に送られ、データ・バスに出力されます (図 6.14)。

このような用途では、一般に雑音環境が良くないので、伝送距離に応じた駆動能力や終端条件を考慮する必要があります。この例では TTL のバス・ドライバである 74LS540 をライン・ドライバ/レシーバとして使っていますが、距離がさらに長くなる場合は差動にするなどの方法を探ります。

次に各信号はなるべくスタティックな状態で伝送するようにします。この例ではアドレス ($A_0 \sim A_1$) をラッチしてから伝送するとともに、 \overline{RD} や \overline{WR} 等のパルス状のストローブ信号はケーブルには乗せていません。

第3に、信号線を減らすよう配慮することです。たとえば、直列転送を使えばソフト/ハードは複雑ですが信号線としては少なくなります。

6.5 分散処理のための直流モータ制御用基板

直流サーボ・モータの制御 (速度, 加速度, 位置決めなど) をマイコンで行うのは結構面倒です。これは、ある程度のリアルタイム性が必要な上、数値計算を伴うためです。ところが、多軸のロボットなどで複数のモータを同時に制御しなければならないことが増えています。そこで、分散処理を行います。

図 6.14 データ入力時の信号の流れ

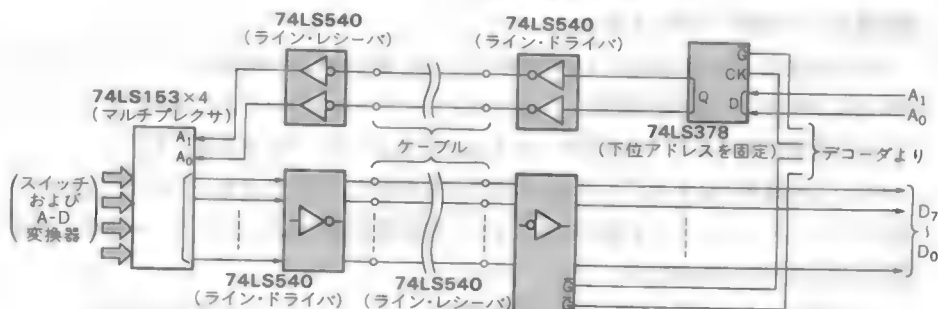
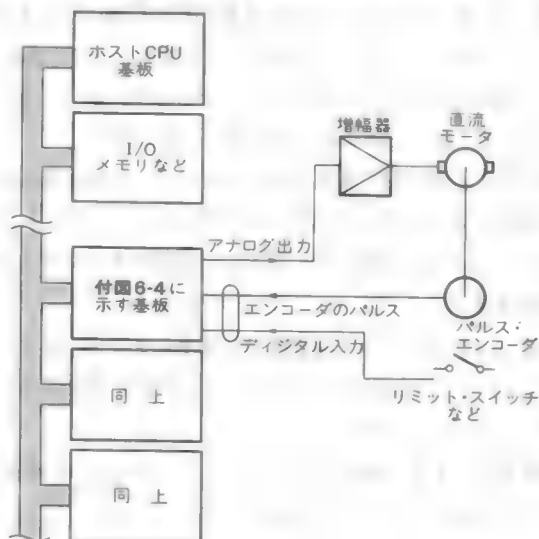


図 6.15 多軸サーボ・システムの構成例



分散処理にも様々な形態が考えられますが、ここで紹介するユニットは図6.15のような構成で使われます。このような構成にすると1個のCPUでモータを制御するのに比べてハードウェアが複雑になるように思えます。しかし、各モータ制御ユニットはすべて同じものであり、モータ数の増減などに対して柔軟性があります。

それにも増して重要なのはソフトウェア上の利点です。すなわち、各モータの特性の違いやモータ数の変化に対して、ホストCPUのソフトウェアはほとんど関知する必要がありません。したがって、大規模であっても柔軟性に富んだシステムが作れます。

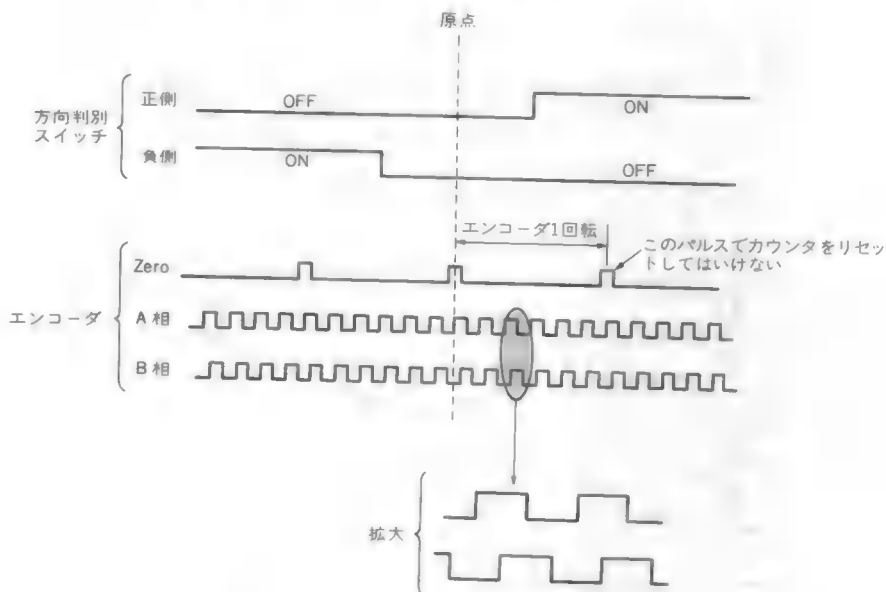
●直流モータ制御に必要な入出力

一般に直流モータの位置決めサーボ機構では、直流モータへの駆動用出力（アナログ電圧またはPWM：パルス幅変調波）とともに、パルス・エンコーダまたはポテンショメータからの位置情報の入力（前者はデジタル、後者はアナログ）が必要です。

また、このほかに若干のデジタル入出力が必要なことがあります。これは、危険防止用のリミット・スイッチや手動の停止スイッチの状態を読み込んだり、リレーやランプの制御を行ったりするためです。

ここで紹介するユニットでは、モータ駆動用として±6Vのアナログ電圧を出力し、モ

図 6.16 エンコーダの出力と方向判別用スイッチとの関係



ータを実際に駆動する電力増幅器は外付けとします。

また、位置検出用としてはパルス・エンコーダを使います。パルス・エンコーダからは1回転に1個のリセット・パルスが出力されますが、これを多回転で用いる場合、カウンタのリセットを防止する必要があります。このために方向判別用の2個の接点信号を入力できるようになっています(図6.16を参照)。

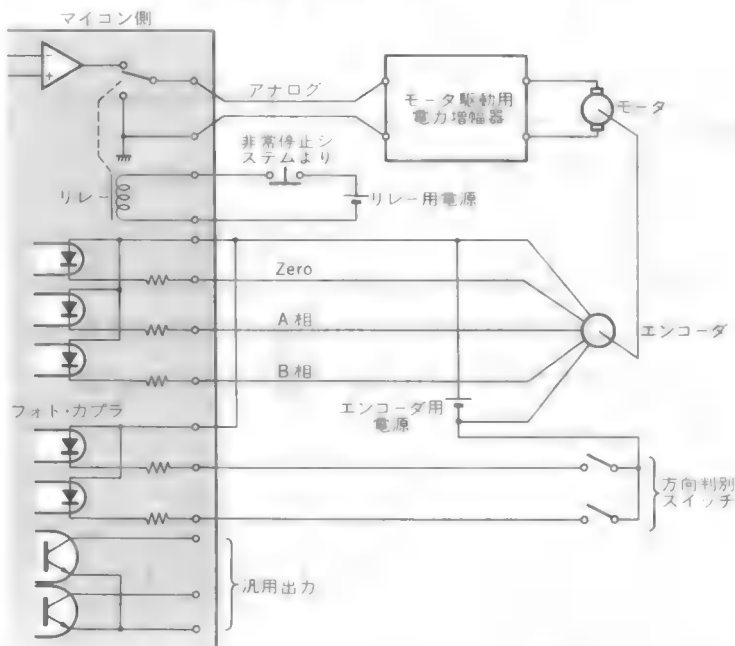
このほか、フォト・カプラで絶縁された2ビットの出力、外部からアナログ出力をオン/オフするためのリレーなどを備えています。後者は万一の非常停止などの際に使用します(図6.17)。

● CPU の選定

この基板の回路を付図6.4に示します。一見したところ、Z80を中心とした独立したマイコン基板のように見えます(事実、単独で動かすことも可能です)が、前述のようにスレーブとして動くよう設計されたものです。

ただし、このように入出力やメモリがきぎられており拡張の必要のない用途には、Z80の

図 6.17 直流モータ制御用入出力の例



ような汎用 CPU は適切ではありません。一般には、8041A や 8048/51 系、F8、6801、Z8 などのシングルチップ・コンピュータを使い、部品数を削減すべきです。

しかし、ソフトウェアの開発環境や部品管理上の問題からハードウェアの統一を要求される場合には、ホストもスレーブも Z80 といった事態が生じるのです。

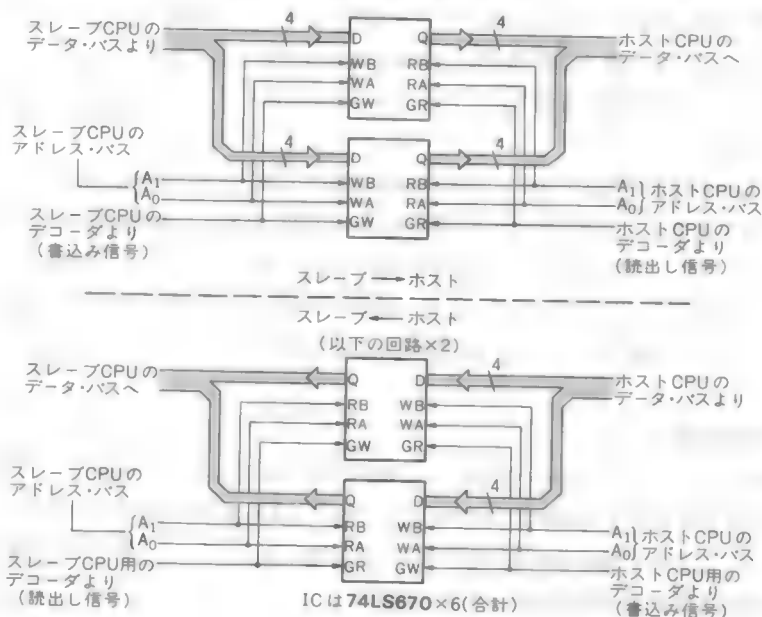
■ホストとの間のデータ交換

CPU として 8041A や Z8 を使えば、ホストは直接にスレーブ CPU の内部レジスタをアクセスできます。しかし、Z80 CPU ではそうはいきませんから、何らかのデータ交換方法を設けなければなりません。

この基板の用途ではホストからスレーブに渡すデータは 4 バイト（サーボ・モータの停止目標位置 16 ビット、速度・加速度のパラメータ各 8 ビット）、スレーブからホストに渡すデータは 3 バイト程度（現在位置 16 ビットと若干のステータス・フラグ）です。

そこで、この回路では IC₁₀～IC₁₅ (74LS670) を使い、ホスト→スレーブは 8 バイト、ス

図 6.18 データ交換レジスタ部



スレーブ→ホストは4バイトのデータ交換レジスタ群としています(図6.18)。このレジスタ群はホスト/スレーブのいずれから見てもメモリ・マップされた入出力ポートに見えます。部品数を少なくするため、出力したデータの読み戻しはできません。

データ交換用としては、このほかにFIFOを使う方法(第5章、例5-4を参照)や直列通信を用いる方法、があります。いずれも非同期的に大量のデータに受け渡しするには適していますが、少量・高速のデータ交換には向きません。

また、この基板ではデータ交換レジスタとしてTTLのレジスタである74LS670(4語×4ビット)を用いています。これは、設計時点(1983年)で適当なデュアル・ポートRAMやバス・レジスタが入手できなかったためです。74LS670は速度の点では充分ですが、1個あたりの容量が少なく消費電流が多いなどの欠点があります。現在ではこれに相当するCMOSのレジスタ、または両側から読み書きできるデュアル・ポートRAMが入手できるので検討をおすすめします。

■データ交換のタイミング

付図 6.4 中、④点の TMG 信号は、ホスト CPU とのデータ交換タイミングを管理するためのものです。この例では約 200Hz (5 ms) の方形波をこの信号として使っています (図 6.19)。マルチ CPU 間の通信方法としては、このようにデータ方向を決めてしまうのがもっとも簡単です。

ただし、この方法ではシステム全体の動作周期が TMG 信号の周期に押えられてしまう欠点があります。この例ではモータ制御に 5 ms の遅れがあっても特に問題とはなっていません。しかし、より高速のデータ交換が必要な場合、図 6.19 のように外部的な同期信号を使うのではなく、セマフォ・ビット (一種のハンドシェイク用のフラグ) を設けて両側からのアクセスが衝突しないように制御する必要があります。

■その他の入出力

付図 6.4 中の IC₂₉ ~ IC₃₂ はパルス・エンコーダ用の 16 ビットの可逆カウンタです。毎秒 100 パルス程度の速度であれば、ソフトウェアでパルスをカウントするのも不可能ではありません。しかし、様々な用途に使われることを考えると、ソフトウェアで複雑なタイミング管理を行わせるのは適切ではありません。

ここで使用するようなインクリメント型のエンコーダでは、電源を投入してから原点を通過するまでは正確な位置がわかりません。そこで、IC₂₈ から成る RS フリップフロップを使い、原点を通過したか否かを覚えておきます。

IC₂、IC₃ は 12 ビットの D-A 変換器です (図 6.20)。ここでは上位 8 ビットをデータ・バス、下位 4 ビットをアドレス・バスに各々接続し、一度に 12 ビットの実行ができるよう

図 6.19 CPU 間のデータ交換タイミング

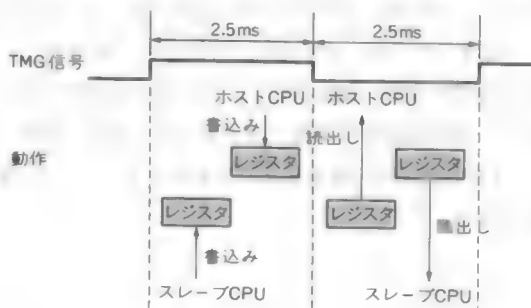


図 6.20 D-A 変換部

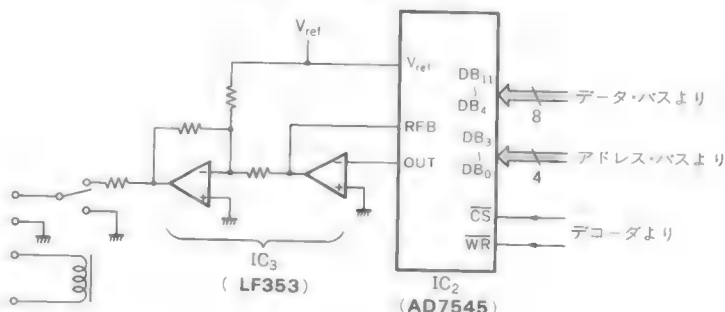
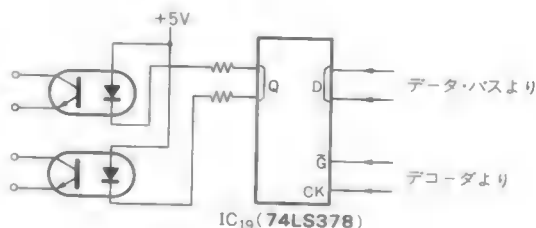


図 6.21 デジタル出力部



になっています。

IC₃からのアナログ出力はリレー接点を介して基板外に導かれます。ここに入っているリレーは、電源投入時などアナログ出力が不定の期間にアナログ電圧をモータに印加しないよう遮断したり、非常停止の■の制御に使ったりするためのものです。

このほか、若干のデジタル入出力がありますが、基板外に引き出す可能性のあるものはすべてフォト・カプラで絶縁してあります。一般に、近くで複数のモータが回っているような場所は、雑音環境がかなり悪いと考えられます。また、デジタル入出力の相手側はリミット・スイッチやシーケンサなど、何をつながれるかわかりません。そこで、最も無難な方法としてフォト・カプラで絶縁しておきます。

デジタル入力のほうは、エンコーダ用カウンタや基板上のDIPスイッチなどとともにIC₂₀～IC₂₃（マルチプレクサ）を使ってデータ・バスに接続してあります（図6.21）。一方、デジタル出力はIC₁₉（Dフリップフロップ）から取り出しています。このへんは第5章で述べた通りの基本的なものです。

Appendix

マイコン・システムのデバッグの例

—— Z80用モニタ・プログラム ——

1. バグをなくすために

万一（残念ながら万に数千の確率で起こるのですが）作った装置が思い通りの動作をしなかった場合、その原因を追求しなければなりません。その装置がマイコンを含んでいる場合には、ソフト/ハードのどちらに原因があるかわからないと手の下しようがありません。これがマイコン・システムのデバッグの面倒さにつながっています。

では、どのようにすればバグの少ない（あるいは見つけやすい）システムになるかを考えてみます。

●設計の注意点

極力簡単なハードウェアを設計すべきです。ただしこれは単に部品数が少ないことを意味するわけではありません。たとえば簡単な入出力ポートを TTL や CMOS の MSI で作った場合と、並列インターフェース用の LSI (Z80 PIO や 8255 など) を使った場合とを比較すると、前者のほうが一見複雑に思われます。

しかし、後者はソフトウェアでの初期設定をしないと動かせないことなどを考えると、必ずしもバグが取りやすいとはいえません。

■製作上の注意点

手作りの装置に見られる欠陥は、ハード/ソフトともに大部分が製作時の人為的ミスといって良いでしょう。それも、ハンダ付けの不良とか、ジャンプ先の間違いなど、いたって初歩的なものが多いようです。

表A 実行できるコマンド例

G [$\langle \text{Add} \rangle$] ✓	$\langle \text{Add} \rangle$ 番地へ実行を移す。 $\langle \text{Add} \rangle$ が省略された場合は0とみなされる。 $\langle \text{Add} \rangle$ を4桁以上入力したときは最後の4桁が有効となる。
R ✓	RST 38H でブレークしたアドレスにリターンする。表のレジスタは復改される。ユーザ・ルーチン内で RST 38H でブレークした時には、その命令をもとにもどしてからリターンしなければならない。
M [$\langle \text{SAdd} \rangle$], [$\langle \text{EAdd} \rangle$], [$\langle \text{TAdd} \rangle$] ✓	$\langle \text{SAdd} \rangle$ から $\langle \text{EAdd} \rangle$ までのメモリの内容を $\langle \text{TAdd} \rangle$ からに移動する。
D [$\langle \text{SAdd} \rangle$], [$\langle \text{EAdd} \rangle$] ✓	$\langle \text{SAdd} \rangle$ から $\langle \text{EAdd} \rangle$ までのメモリの内容を表示する。 $\langle \text{SAdd} \rangle$ を省略するとその前に使用したアドレスの値になり、 $\langle \text{EAdd} \rangle$ を省略すると、 $\langle \text{SAdd} \rangle + 3\text{FH}$ となる。
S [$\langle \text{Add} \rangle$] ✓	$\langle \text{Add} \rangle$ を入力すると、その番地の内容を表示して止まるので変更する時はその値2バイトを入力し、変えない時はスペースを入力する。アドレスをもどすときは Back Space を入力する。 $\langle \text{Add} \rangle$ を省略するとその前に使用したアドレスの値になる。
F $\langle \text{SAdd} \rangle$, $\langle \text{EAdd} \rangle$, $\langle \text{CONST} \rangle$	$\langle \text{SAdd} \rangle$ から $\langle \text{EAdd} \rangle$ までの内容を $\langle \text{CONST} \rangle$ でうめる。各アドレスとデータは省略できない。
X	'X' を入力するとブレークした時の PC, AF, BC, DE, HL, SP の内容を表示する。
I [$\langle \text{Add} \rangle$] ✓ I [$\langle \text{Add} \rangle$],	$\langle \text{Add} \rangle$ ポート番地の内容を表示する。省略した場合は0となる。',' の場合は次の行に、',' の場合はその行に内容を表示する。
O [$\langle \text{Add} \rangle$], $\langle \text{Data} \rangle$ ✓	$\langle \text{Add} \rangle$ ポート番地に $\langle \text{Data} \rangle$ を出力する。

しておくことです。ただし、ソフトウェアもハードの構成によって変更する必要があります。したがって、少なくとも一部の入出力のアドレス配置は統一しておくと便利です。

ここで紹介するモニタ・プログラムは図1(a)または(b)のハードウェアを前提としています。このうち(a)のほうは■6章のCPU基板で使用できます。また、(b)のほうはCTCや8251Aを持たないシステムであっても、I/O領域に入出力ポート(Z80 PIOまたはハードウェアによるもの)があれば使用できます。ただし、いずれもクロック周波数とボーレートとが関係するので、クロック周波数を変更する際は若干の修正が必要です。

このモニタ・プログラムで実行できるコマンドは表Aの通りです。これは日本電気のPC8801の内蔵のモニタ・プログラム、あるいはCP/M上で走るSID(シンボリック・デバグ)と良く似ていますので、これらを開発装置として使う■にも便利です。

なお、この程度の簡単なモニタ・プログラム(198ページ参照)は、使用するCPUを理解するのに最適です。したがって、初めて使用するCPUには自分でモニタ・プログラムを作ってみると、ハード/ソフトともに参考となるでしょう。

III. デバッグの実際

作った装置をデバッグするには、前述のモニタ・プログラムを書き込んだROMを実装し、充分な検査の終わったハードウェアにターミナル(またはパーソナル・コンピュータをターミナル・モードとしたもの)を接続し、電源を投入すれば良いわけです。するとターミナルの画面に、

```
RAM AREA 1000 1FFF
```

*

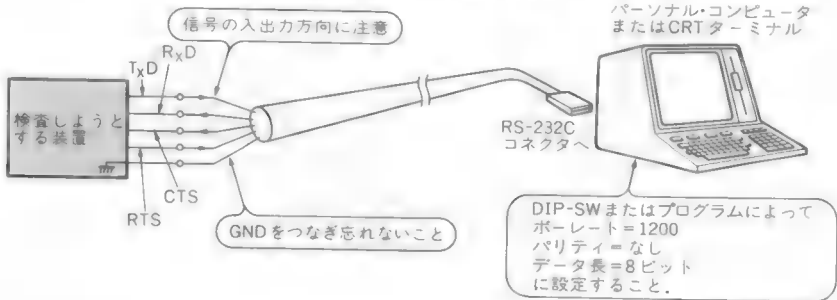
と表示が出るでしょう。出た場合は次項を飛ばして197ページをご覧ください。しかし万一(たぶん万に数千の率ですから悲観しないように……) 出ない場合は、以下を良くお読み下さい。

■ハードウェアのデバッグ

モニタ・プログラムが走らない場合、ハードウェアに何らかの異常があると考えられます。ただし、その前に次の点を確認して下さい。

- (1) ターミナル(またはパーソナル・コンピュータ)との直列通信回■の設定に間違いは

図II 直列通信回線の接続



ないか？たとえばRS-232C コネクタの入出力の違い、レベル(±12V か TTL レベルか)はどうか、パリティやストップ・ビット、信号の正負の違い、CTS や RTS のレベルの違い、ボーレート(前述のモニタ・プログラムは1200BPS用)の違いなど(図IIを参照)。

(2) プロンプト(*)が連続して出続ける場合はターミナルからスペースか **CR** (キャリッジ・リターン) を送ってみる。

(3) 何も出ない場合、作った装置のCPUに念のためリセットをかけてみる。

以上の3点で解決すれば、ハードウェアの異常ではありませんので以下を飛ばして196ページからご覧下さい。これで解決しない場合は、いよいよハードウェアの検査が必要になります。

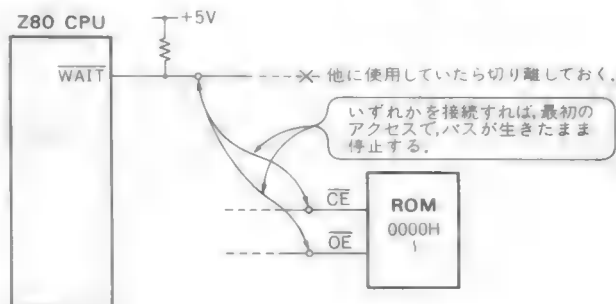
ハードウェアの本格的な検査には2現象以上のオシロスコープが必要です。ただし、場合によってはテストだけで見当をつけなければならないこともあるでしょうから、そのような場合も考慮して話を進めます。

ハードウェアの検査といっても様々な方法があるでしょうが、たとえば次のような手順で行います。

- (1) 各ICには所定の電源が供給されているか？
- (2) クロック発振が行われているか？
- (3) CPUは動いているか？

これはZ80 CPUの $\overline{M1}$ 出力をオシロスコープで見るのが簡単です。

以上がOKでしたらハードウェアの致命傷はありません。次にもう少し本格的な検査を

図III CPUの $\overline{\text{WAIT}}$ 入力を使う方法図IV モニタ・プログラムの使用例(下線部を使用者が打ち込む。へは $\boxed{\text{CR}}$)

* S F F 0 0 Δ
 F F 0 0 F F 0 0 F F 0 0番地への書き込みデータ
 ? 読み飛ばせないと?が出る
 * S 次回はアドレス不要
 F F 0 0 F F - 以下、繰り返して必要なデータを書き込み、確認する

(a) メモリ領域 0 F F 0 0番地の出力ポートの検査

* I 8 0 2 0 番地のあとに・を打つと改行なしにデータを読み込む
 * I 8 0 2 1 入力状態を変えて必要な回数の検査を繰り返す
 * I 8 0 2 2

(b) I/O領域 8 0番地の入力ポートの検査

行います。

(4) CPUの $\overline{\text{HALT}}$ 出力を調べる。前述のモニタ・プログラムでは、RAMが実装されていないと判断すると $\overline{\text{HALT}}$ となります。したがって、電源投入後、1秒かそこら(リセット回路等により異なる)で $\overline{\text{HALT}}$ が“L”になるようなら、RAMの非実装か配線の異常です。また、モニタ・プログラム自体が非実装だとCPUがFFH(RST 38H)を2回実行して $\overline{\text{HALT}}$ となることがありますから、ROMの配線ミスも調べます。

(5) ROMの $\overline{\text{OE}}$ をCPUの $\overline{\text{WAIT}}$ 入力に接続してバスの状態を調べます。上記(4)まで異常が見られないのなら、バスの異常(ショートや開放など)を調べます。図IIIに示すようにCPUの $\overline{\text{WAIT}}$ 入力を使う方法は、テスト1台しか手許にない場合にも非常に有効で簡単に実行できるので、是非覚えておいて下さい。

たとえば、ROMの $\overline{\text{OE}}$ を接続すれば最初にROMを読み出した瞬間に $\overline{\text{WAIT}}$ がかかり、アドレス・バスは0000H、データ・バスはその内容になるはずで、同様にして他

の I/O 用の IC やメモリの \overline{CE} , \overline{OE} , \overline{WE} , あるいはアドレス線をうまく \overline{WAIT} とつなげば、各バスの状態がスタティックで（したがってテスト 1 台でも）観察でき、非常に役に立ちます。

設計が正しく、部品に異常がなく、しかも配線に大きなミスがなければ、以上の検査で大半のミスが見つかるものです。ここまででミスが発見できない場合、あせらずにもう一度、配線の確認をおすすめします。

配線のミスは慣れた人でも 1%～数%ありますが、部品の不良はそれより 1 桁以上少ないのです。

●モニタ・プログラムによる各部の検査

いよいよモニタ・プログラムが動いたら、これを使って各部の検査を行います。たとえば、メモリ領域への書込み・読出しには S コマンドを使います。この場合、書き込んだ内容が同じアドレスで読み出せないと、? を表示しますが異常ではありません。

また、I/O 領域は I コマンドおよび O コマンドを用いますが、こちらは同じアドレスで読み返せなくても構いません。

この様子を図 IV に示すので参考として下さい。

<リスト> Z80 用モニタ・プログラム

```

;
;      OKmon Ver 1.2 (PIO Version)
;
;      OK monitor Version 1.2
;      PIO Version
;      1983.03.08
;

0000'      ASEG
           .Z80

0000      PIO      EQU      0

;
ENTRY:    ORG      0
          LD        A,DCFH
          OUT       (PIO+2),A      ;PIO MODE 3 SET
          LD        A,0BEH
          OUT       (PIO+2),A
          LD        A,03H
          OUT       (PIO+2),A      ;PIO DI
          LD        A,0FFH
          OUT       (PIO),A
          JP        MSRCH

;
COMP:     ORG      30H
          PUSH      HL
          AND       A
          SBC       HL,DE
          POP       HL
          RET

;
RST38:    ORG      38H
          EX        (SP),HL
          DEC       HL
          EX        (SP),HL
          PUSH      AF
          PUSH      BC
          PUSH      DE
          PUSH      HL
          LD        HL,0AH
          ADD       HL,SP
          PUSH      HL
          DEC       HL
          LD        D,(HL)
          DEC       HL
          LD        E,(HL)
          EX        DE,HL
          CALL      LINOUT
          DEFB      0DH,0AH
          DEFB      ' '
          DEFB      0FFH
          CALL      SEIN

;
CP        'G'
JR        NZ,RTN

;
GUSR:     CALL      GETAD
          JR        NZ,ERR
          CALL      CRLF
          EX        DE,HL
          POP       HL
          LD        SP,HL
          EX        DE,HL
          JP        (HL)

;
NM1:      ORG      66H
          EXX
          POP       DE
          ;NM1 ENTRY

```

0068	21 006E		LD	HL, NM11	
0068	E5		PUSH	HL	
006C	ED 45		RETN		: IFF2 -> IFF1
006E	D5	NM11:	PUSH	DE	
006F	D9		EXX		
0070	18 C6		JR	RST38	
0072	FE 52	;			
0074	20 19	RTN:	CP	'R'	
			JR	NZ, MON1	
0076	CD 012F	;	CALL	SEIN	: RETURN FROM RST38
0079	FE 00		CP	0DH	
007B	20 09		JE	NZ, ERR	
0070	CD 008B		CALL	CRLF	
0080	E1		POP	HL	
0081	E1		POP	HL	: RECOVER REG.
0082	D1		POP	DE	
0083	C1		POP	BC	
0084	F1		POP	AF	
0085	C9		RET		
0086	CD 030A	ERR:	CALL	LINOUT	: ERROR
0089	DD 0A		DEFB	0DH, 0AH	
008B	3F		DEFB	' '	
008C	FF		DEFB	0FH	
008D	18 BA		JR	MON	
008F	FE 40	MON1:	CP	'M'	
0091	CA 01DF		JP	Z, MEM	: MOVE MEMORY
0094	FE 44		CP	'D'	
0096	CA 0212		JP	Z, DMP	: DUMP MEMORY
0099	FE 53		CP	'S'	
009B	CA 0258		JP	Z, SETM	: SET MEMORY
009E	FE 46		CP	'F'	
00A0	CA 02BE		JP	Z, FILM	: FILL MEMORY WITH CONSTANT
00A3	FE 58		CP	'X'	
00A5	CA 0316		JP	Z, XMINR	: EXAMINE REGISTER
00A8	FE 49		CP	'I'	
00AA	CA 02E0		JP	Z, INP	: INPUT PORT
00AD	FE 4F		CP	'O'	
00AF	CA 02F5		JP	Z, OUTP	: OUTPUT PORT
00B2	18 95		JR	MON	
00B4	DB 00	BREAK:	IN	A, (PIO)	: BREAK CHECK
00B6	E4 80		AND	80H	
00B8	07		RLCA		
00B9	3F		CCF		
00BA	C9		RET		
00BB	3E 00	;			
00BD	CD 00E3	CRLF:	LD	A, 0DH	
00C0	3E 0A		CALL	SOUT	
00C2	18 1F		LD	A, 0AH	
			JR	SOUT	
00C4	3E 20	;			
00C6	18 1E	SPC:	LD	A, ' '	
			JR	SOUT	
00C8	7E	;			
00C9	18 05	DOUT:	LD	A, (HL)	: (HL) HEX OUT
			JR	HXOUT	
00CB	7C	;			
00CC	CD 00D0	AOUT:	LD	A, H	: HL HEX OUT
00CF	7D		CALL	HXOUT	
			LD	A, L	
00D0	F5	;			
00D1	0F	HXOUT:	PUSH	AF	: HEX BYTE OUT
00D2	0F		RRCA		
00D3	0F		RRCA		
00D4	0F		RRCA		
00D5	CD 00D9		CALL	HOUT1	
00D8	F1		POP	AF	
00D9	E4 0F	HOUT1:	AND	0FH	
00DB	FE 0A		CP	0AH	
00DD	38 02		JR	C, HOUT2	
00DF	C6 07		ADD	A, 7	
00E1	C6 30	HOUT2:	ADD	A, 30H	
		;			

```

00E3 F3          SOUT:  DI
00E4 F5          PUSH
00E5 08          EX      AF,AF'      ;OUTPUT DATA SAVE
00E6 08 00      SOUT1: IN      A,(PIO)      ;OUTPUT DATA SAVE
00E8 CB 4F      BIT      1,A          ;CTS ON ?
00EA 20 FA      JR      NZ,SOUT1
00EC E6 FE      AND      0FEH
00EE 03 00      OUT      (PIO),A      ;START BIT OUTPUT
00F0 09          EXX
00F1 21 011F    LD      HL,CNST1
00F4 CD 0183    CALL    BSET
00F7 CD 0192    CALL    DELAY1      ;START BIT DELAY
00FA CD 018B    CALL    BSET1      ;DELAY CONST.->DE
00FD DB 00      IN      A,(PIO)
00FF E6 72      AND      72H
0101 6F          LD      L,A          ;USER DATA->L
0102 08          EX      AF,AF'
0103 4F          LD      C,A          ;OUTPUT DATA->C
0104 06 09      LD      B,9
0106 00          NOP
0107 37          SCF
0108 7D          SOUT2: LD      A,L
0109 CB 19      RR      C            ;1 BIT->CY
010B CE 00      ADC      A,0
010D 03 00      OUT      (PIO),A      ;1 BIT OUTPUT
010F CD 0192    CALL    DELAY1      ;FULL BIT DELAY
0112 10 F4      DJNZ    SOUT2
0114 06 07      LD      B,7
0116 10 FE      DJNZ    SOUT3
0118 CD 0195    CALL    DELAY2
011B F1          POP      AF          ;RECOVER ACC
011C 09          EXX              ;RECOVER REG.
011D FB          EI
011E C9          RET

;
011F 0397      CNST1: DEFW    919      ;110 BPS
0121 039F      DEFW    927

;
0123 0149      DEFW    329      ;300 BPS
0125 0151      DEFW    337

;
0127 0049      DEFW    73       ;1200 BPS
0129 0051      DEFW    81

;
012B 0009      DEFW    9        ;4800 BPS
012D 0011      DEFW    17

;
012F C5          SEIN:  PUSH    BC
0130 08 00      IN      A,(PIO)
0132 E6 31      AND      31H
0134 03 00      OUT      (PIO),A      ;RTS ON
0136 E6 30      AND      30H
0138 08          EX      AF,AF'      ;USER DATA SAVE
0139 09          EXX
013A 00          SEIN1: IN      A,(PIO)
013C E6 80      AND      80H
013E C2 013A    JP      NZ,SEIN1      ;START BIT ?
0141 21 019E    LD      HL,CONST
0144 CD 0183    CALL    BSET
0147 CD 0195    CALL    DELAY2      ;HALF BIT DELAY
014A CD 018B    CALL    BSET1      ;DELAY CONST.->DE
014D 00          IN      A,(PIO)
014F CB 7F      BIT      7,A
0151 20 E7      JR      NZ,SEIN1      ;START BIT ?
0153 00          EX      AF,AF'
0154 6F          LD      L,A          ;USER DATA->L
0155 03 00      OUT      (PIO),A      ;START BIT ECHO BACK
0157 03          INC      BC          ;NOP 6T
0158 06 08      LD      B,B
015A CD 0190    CALL    DELAY      ;FULL BIT DELAY
015D 08 00      SEIN2: IN      A,(PIO)
015F E6 80      AND      80H
0161 07          RLCA
0162 CB 19      OR      C
0164 85          LD      L,A          ;USER DATA
0165 03 00      OUT      (PIO),A      ;ECHO BACK
0167 CD 0190    CALL    DELAY      ;FULL BIT DELAY

```



```

016A 10 F1          DJNZ SEIN2
016C 06 02          LD B,2
016E 10 FE          SEIN3: DJNZ SEIN3          ;TIME DELAY
0170 3E 41          SEIN4: LD A,41H
0172 B5             OR L
0173 03 00          OUT (PIO),A          ;STOP BIT OUTPUT
0175 06 07          LD B,7
0177 10 FE          SEIN5: DJNZ SEIN5          ;TIME DELAY
0179 CD 0190        CALL DELAY
017C CD 0190        CALL DELAY
017F 79             LD A,C          ;DATA->A
0180 D9             EXX
0181 C1             POP BC
0182 C9             RET

;
0183 DB 00          BSET: IN A,(PIO)          ;BAUD RATE->DE
0185 E6 0C          AND 0CH
0187 4F             LD C,A
0188 06 00          LD B,0
018A 09             ADD HL,BC
018B SE             LD E,(HL)
018C 23             INC HL
018D S6             LD D,(HL)
018E 23             INC HL
018F C9             RET

;
0190 A7             DELAY: AND A          ;NOP 4T
0191 08             RET C          ;NOP 5T
0192 3A 0000        DELAY1: LD A,(0)          ;NOP 13T
0195 05             DELAY2: PUSH DE
0196 18             DELAY3: DEC DE
0197 7A             LD A,D
0198 B3             OR E
0199 C2 0196        JP NZ,DELAY3          ;DE=0 ?
019C D1             POP DE
019D C9             RET

;
019E 01C8          CONST: DEFW 456          ;110 BPS
01A0 039E          DEFW 926

;
01A2 00A1          DEFW 161          ;300 BPS
01A4 0150          DEFW 336

;
01A6 0021          DEFW 33          ;1200 BPS
01A8 0050          DEFW 60

;
01AA 0001          DEFW 1          ;4800 BPS
01AC 0010          DEFW 16

;
01AE 06 30          CKHEX: SUB 30H
01B0 08             RET C
01B1 FE 0A          CP 0AH
01B3 38 07          JR C,CKHX1          ;0-9
01B5 06 11          SUB 11H
01B7 08             RET C          ;:-?
01B8 C6 0A          ADD A,0AH
01BA FE 10          CP 10H
01BC 3F             CKHX1: CCF
01BD C9             RET

;
01BE 21 0000        GETAD: LD HL,0          ;GET ADDRESS -> HL
01C1 CD 012F        GTAD1: CALL SEIN
01C4 FE 0D          CP 0DH
01C6 C8             RET Z          ;CR .. Z=1
01C7 FE 2C          GTAD2: CP ','
01C9 20 03          JR NZ,GTAD3
01CB D6 00          SUB 0
01CD C9             RET          ;',',... Z=0
01CE CD 01AE        GTAD3: CALL CKHEX
01D1 30 04          JR NC,GTAD4
01D3 F1             POP AF          ;DUMMY POP
01D4 C3 0086        GTAD4: JP ERR
01D7 29             ADD HL,HL
01D8 29             ADD HL,HL
01D9 29             ADD HL,HL
01DA 29             ADD HL,HL
01DB 85             ADD A,L

```

010C	6F		LD	L,A	
010D	18 E2		JR	GTAD1	
010F	CD 018E	I			
01E2	CA 0086	MEM:	CALL	GETAD	: 'H' COMMAND
01E5	EB		JP	Z,ERR	
01E6	CD 018E		EX	DE,HL	:FROM START ADDRESS
01E9	CA 0086		CALL	GETAD	:FROM END ADDRESS
01EC	A7		JP	Z,ERR	
01ED	ED 52		AND	A	
01EF	DA 0086		SBC	HL,DE	
01F2	23		JP	C,ERR	:DE>HL .. ERROR
01F3	44		INC	HL	
01F4	4D		LD	B,H	:BYTES = BC
01F5	CD 018E		LD	C,L	
01F8	C2 0086		CALL	GETAD	:TO ADDRESS
01F8	ES		JP	NZ,ERR	
01FC	EB		PUSH	HL	
01FD	F7		EX	DE,HL	:FROM(HL) TO(DE) BYTES(BC)
01FE	38 06		RST	30H	:CALL COMP
0200	ED 80		JR	C,MEM1	
0202	E1		LDIR		
0203	C3 0049		POP	HL	
			JP	MON	
0206	09	I			
0207	28	MEM1:	ADD	HL,BC	:DE>HL
0208	EB		DEC	HL	
0209	09		EX	DE,HL	
020A	28		ADD	HL,BC	
020B	EB		DEC	HL	
020C	ED 88		EX	DE,HL	
020E	E1		LDDR		
020F	C3 0049		POP	HL	
			JP	MON	
0212	CD 012F	I			
0215	FE 0D	DMP:	CALL	SEIN	:DUMP MEMORY
0217	28 19		CP	0DH	
0219	FE 2C		JR	Z,DMP2	
021B	28 08		CP	,,	
021D	21 0000		JR	Z,DMP1	
0220	CD 01CE		LD	HL,0	
0223	28 0D		CALL	GTAD3	:START ADDRESS
0225	EB		JR	Z,DMP2	
0226	CD 018E	DMP1:	EX	DE,HL	
0229	C2 0086		CALL	GETAD	:END ADDRESS
022C	F7		JP	NZ,ERR	
022D	DA 0086		RST	30H	
0230	18 05		JP	C,ERR	
0232	EB		JR	DMP3	
0233	21 003F	DMP2:	EX	DE,HL	
0236	19		LD	HL,3FH	
0237	EB		ADD	HL,DE	
0238	CD 0088	DMP3:	EX	DE,HL	
023B	CD 00CB	DMP4:	CALL	CRLF	
023E	CD 00C4		CALL	AOUT	:ADDRESS OUT
0241	CD 00C4		CALL	SPC	
0244	CD 00C8	DMP5:	CALL	SPC	
0247	CD 0084		CALL	DOUT	:DATA OUT
024A	DA 0049		CALL	BREAK	:BREAK CHECK
024D	23		JP	C,MON	
024E	EB		INC	HL	
024F	F7		EX	DE,HL	
0250	EB		RST	30H	
0251	DA 0049		EX	DE,HL	
0254	7D		JP	C,MON	: HL > DE
0255	E6 0F		LD	A,L	
0257	2D E8		AND	0FH	
0259	18 0D		JR	NZ,DMP5	
			JR	DMP4	
025B	CD 012F	I			
025E	FE 0D	SETM:	CALL	SEIN	:SET MEMORY
0260	28 09		CP	0DH	
0262	21 0000		JR	Z,STM1	
0265	CD 01CE		LD	HL,0H	
0268	C2 0086		CALL	GTAD3	
026B	CD 008B	STM1:	JP	NZ,ERR	
			CALL	CRLF	

026E	CD 00CB		CALL	AOUT	
0271	CD 00C4		CALL	SPC	
0274	CD 00C4	STM2:	CALL	SPC	
0277	CD 00CB		CALL	DOUT	
027A	3E 2D		LD	A, ' '	
027C	CD 00E3		CALL	SOUT	
027F	CD 012F		CALL	SEIN	
0282	FE 0D		CP	DDH	
0284	CA 0049		JP	Z, MON	:END
0287	FE 08		CP	DBH	
0289	2B 2B		JR	Z, STM5	:BACK SPACE .. ADDRESS DEC.
028B	FE 20		CP	' '	
028D	20 05		JR	NZ, STM3	
028F	CD 00C4		CALL	SPC	:SPACE .. NO CHANGE
0292	1B 1A		JR	STM4	
0294	CD 01AE	STM3:	CALL	CKHEX	
0297	DA 00B6		JP	C, ERR	
029A	07		RLCA		
029B	07		RLCA		
029C	07		RLCA		
029D	07		RLCA		
029E	4F		LD	C, A	
029F	CD 012F		CALL	SEIN	
02A2	CD 01AE		CALL	CKHEX	
02A5	DA 00B6		JP	C, ERR	
02A8	81		ADD	A, C	
02A9	77		LD	(HL), A	
02AA	8E		CP	(HL)	:WRITE CHECK
02AB	C2 00B6		JP	NZ, ERR	
02AE	23	STM4:	INC	HL	:ADDRESS INC.
02AF	7D		LD	A, L	
02B0	E6 07		AND	07H	
02B2	2B B7		JR	Z, STM1	
02B4	1B BE		JR	STM2	
02B6	3E 2D	STM5:	LD	A, ' '	:ADDRESS DEC.
02B8	CD 00E3		CALL	SOUT	
02BB	2B		DEC	HL	
02BC	1B AD		JR	STM1	
02BE	CD 01BE	:			
02C1	CA 00B6	FILM:	CALL	GETAD	:FILL MEMORY WITH CONSTANT
02C4	E8		JP	Z, ERR	
02C5	CD 01BE		EX	DE, HL	:DE = START ADDRESS
02C8	CA 00B6		CALL	GETAD	
02CB	F7		JP	Z, ERR	
02CC	DA 00B6		RST	30H	
02CF	E5		JP	C, ERR	:DE>HL .. ERROR
02D0	CD 01BE		PUSH	HL	
02D3	C2 00B6		CALL	GETAD	
02D6	7D		JP	NZ, ERR	
02D7	E1		LD	A, L	
02D8	77	FLM1:	POP	HL	:HL = END ADDRESS
02D9	F7		LD	(HL), A	
02DA	CA 0049		RST	30H	
02DD	2B		JP	Z, MON	
02DE	1B FB		DEC	HL	
			JR	FLM1	
02E0	EB	:			
02E1	CD 01BE	INP:	EX	DE, HL	:INPUT PORT
02E4	C2 02EA		CALL	GETAD	:CURRENT ADDRESS SAVE
02E7	CD 00B8		JP	NZ, INP1	
02EA	44		CALL	CRLF	
02EB	4D	INP1:	LD	B, H	
02EC	ED 78		LD	C, L	
02EE	CD 00DD		IN	A, (C)	
02F1	E9		CALL	HXOUT	
02F2	C3 0049		EX	DE, HL	:CURRENT ADDRESS RECOVER
			JP	MON	
02F5	EB	:			
02F6	CD 01BE	OUTP:	EX	DE, HL	:OUTPUT PORT
02F9	CA 00B6		CALL	GETAD	:ADDRESS SAVE
02FC	44		JP	Z, ERR	
02FD	4D		LD	B, H	
02FE	CD 01BE		LD	C, L	
			CALL	GETAD	

```

0301 C2 00B6 JP NZ,ERR
0304 ED 69 OUT (C),L
0306 EB EX DE,HL ;ADDRESS RECOVER
0307 C3 0049 JP MON
;
030A E3 LINOUT: EX (SP),HL ;PRINT STRINGS
030B 7E LD A,(HL)
030C 23 INC HL
030D E3 EX (SP),HL
030E FE FF CP OFFH ;FF .. END
0310 C8 RET Z
0311 CD 00E3 CALL SOUT
0314 18 F4 JR LINOUT
;
0316 EB XMNR: EX DE,HL ;EXAMINE REG.
0317 21 000C LD HL,00CH
031A 39 ADD HL,SP
031B CD 030A CALL LINOUT
031E 00 0A DEFB 00H,0AH
0320 20 50 43 20 DEFB ' PC '
0324 20 20 41 20 DEFB ' A '
0328 46 20 20 42 DEFB ' B '
032C 20 43 20 20 DEFB ' C '
0330 44 20 45 20 DEFB ' D E '
0334 20 48 20 4C DEFB ' H L '
0338 20 20 53 50 DEFB ' SP '
033C 00 0A DEFB 00H,0AH
033E FF DEFB OFFH
033F 06 06 LD B,6
0341 28 XMNR1: HL
0342 CD 00C4 CALL SPC
0345 CD 00C8 CALL DOUT
0348 28 DEC HL
0349 CD 00C8 CALL DOUT
034C 10 F3 DJNZ XMNR1
034E EB EX DE,HL
034F C3 0049 JP MON
;
0352 21 0FFF MSRCH: LD HL,0FFFH ;RAM SERCH ROUTINE
0355 06 00 LD B,0 ;B-REG .. FIND FLAG
0357 23 SRCH1: INC HL
0358 7E LD A,(HL)
0359 2F CPL
035A 77 LD (HL),A
035B 8E CP (HL)
035C 20 0F JR NZ,SRCH2
035E 2F CPL
035F 77 LD (HL),A
0360 8E CP (HL)
0361 20 0A JR NZ,SRCH2
0363 78 LD A,B
0364 A7 AND A
0365 20 F0 JR NZ,SRCH1
0367 54 LD D,H ;RAM START ADDRESS -> DE
0368 50 LD E,L
0369 06 FF LD B,0FFH ;FIND FLAG ON
036B 18 EA JR SRCH1
;
036D 78 SRCH2: LD A,B
036E A7 AND A
036F 20 06 JR NZ,SRCH3
0371 7C LD A,H
0372 A5 AND L ;HL=FFFFH ?
0373 3C INC A
0374 20 E1 JR NZ,SRCH1
0376 76 HALT ;NO RAMS
;
0377 F9 SRCH3: LD SP,HL ;STACK POINTER SET
0378 CD 030A CALL LINOUT
037B 00 0A DEFB 00H,0AH
037D 52 41 40 20 DEFB 'RAM '
0381 41 52 45 41 DEFB 'AREA'
0385 20 20 20 DEFB ' '
0388 FF DEFB OFFH
0389 EB EX DE,HL
038A E5 PUSH HL

```

```

0388      CD 00CB          CALL      AOUT          ;RAM START ADDRESS
038E      CD 030A          CALL      LINOUT
0391      20 2E 2E 20      DEFB      '... '
0395      FF               DEFB      0FFH
0396      EB               EX        DE,HL
0397      2B               DEC      HL          ;RAM END ADDRESS
0398      CD 00CB          CALL      AOUT
039B      CD 00BB          CALL      CRLF
039E      E1               POP      HL

;
039F      11 0800          CKSAM: LD      DE,800H
03A2      3A 0FFF          LD      A,(0FFFH)
03A5      FE 55            CP      55H
03A7      20 03            JR      NZ,CKSM1
03A9      11 1000          LD      DE,1000H
03AC      21 0000          CKSM1: LD      HL,0
03AF      AF              XOR      A
03B0      86              CKSM2: ADD     A,(HL)
03B1      23              INC      HL
03B2      F7              RST      30H
03B3      20 FB            JR      NZ,CKSM2
03B5      A7              AND      A
03B6      2B 19            JR      Z,CKSM3
03B8      CD 030A          CALL      LINOUT
03B8      52 4F 4D 20      DEFB      'ROM '
03BF      43 48 45 43      DEFB      'CHEC'
03C3      4B 53 41 4D      DEFB      'KSAM'
03C7      20 45 52 52      DEFB      'ERR'
03C8      4F 52 21        DEFB      'OR!'
03CE      0D 0A FF        DEFB      0DH,0AH,0FFH

;
03D1      7A              CKSM3: LD      A,D
03D2      FE 10            CP      10H
03D4      CA 0800          JP      Z,800H
03D7      FF              RST      38H

END

```

Macros:

Symbols:

00CB	AOUT	00B4	BREAK	0183	BSET
0188	BSET1	01AE	CKHEX	018C	CKHX1
039F	CKSAM	03AC	CKSM1	0380	CKSM2
0301	CKSM3	011F	CNST1	0030	COMP
019E	CONST	00BB	CRLF	0190	DELAY
0192	DELAY1	0195	DELAY2	0196	DELAY3
0212	DMP	0225	DMP1	0232	DMP2
0237	DMP3	0238	DMP4	0241	DMP5
00C8	DOUT	0000	ENTRY	0086	ERR
02BE	FILM	0208	FLM1	018E	GETAD
01C1	GTA01	01C7	GTA02	01CE	GTA03
0107	GTA04	0057	GUSR	00D9	HOUT1
00E1	HOUT2	0000	HXOUT	02E0	INP
02EA	INP1	030A	LINOUT	010F	MEM
0206	MEM1	0049	MON	008F	MON1
0352	MSRCH	0066	NM1	006E	NM11
02F5	OUTP	0000	P10	003B	RST38
0072	RTN	012F	SEIN	013A	SEIN1
0150	SEIN2	016E	SEIN3	0170	SEIN4
0177	SEIN5	0258	SETM	00E3	SOUT
00E6	SOUT1	0108	SOUT2	0116	SOUT3
00C4	SPC	0357	SRCH1	0360	SRCH2
0377	SRCH3	026B	STM1	0274	STM2
0294	STM3	02AE	STM4	0286	STMS
0316	XMINR	0341	XMNR1		

参考文献

- (1) 猪飼國夫：インターフェース回路の設計，1980，CQ出版社
- (2) 野崎 真：インターフェース設計入門，1983，ナツメ社
- (3) Zilog：*Microcomputer Components Data Book*，1981
- (4) Intel：*Component Data Catalog*，1982
- (5) T.I.：*The Bipolar Integrated Circuits Data Book*，1981，*part 1, part 2*
- (6) Analog Devices：*Data-Acquisition Databook*，1982
- (7) Intersil：*Data Book*，1982
- (8) N.S.：*Linear Databook*，1982
- (9) RCA：*COS/MOS Integrated Circuits*，1980
- (10) N.S.：*CMOS Databook*，1981
- (11) 日立：HD46505S ユーザーズ マニュアル
- (12) 日立：HD44780 ユーザーズ マニュアル
- (13) Motorola：*CMOS Data*，1981
- (14) Siliconix：*Analog Switch & IC Product Data Book*，1982
- (15) 松下電工：ナショナル電子回路用リレー，1983
- (16) Intel：*iAPX 88 Book*，1981

索引

【ア 行】

アイソレーション	71
アクイジション時間	86
アクセス時間	31, 32, 35, 52
アドレスブル(ビット指定可能)・ラッチ	105, 106, 113
アドレス・バス	41
アドレス・ラッチ	143
アナログ・グラウンド	89, 137
アナログ出力	166
アナログ・スイッチ	108, 131, 133, 134, 136
アナログ-デジタル変換器	47
アナログ入力	166
映像信号	154
液晶	97, 166
液晶表示器	118, 124
エンコード(符号化)	51, 61, 66, 174
オープン・コレクタ	44, 52, 74, 91, 118, 166, 178
押しボタン・スイッチ	61, 170, 174
折返し歪	85
音声合成	149
音声認識	149

【カ 行】

間接アドレッシング	102
キー・スイッチ	63
グリッチ	100, 108
工業用TVカメラ	158
コントロール・バス	41
コンパレータ	48, 61, 74, 81, 82

【サ 行】

サージ電圧	127
サイクル・スチール	151
サム・ホイール・スイッチ	69, 170, 174

サンプル・アンド・ホールド回路	86
シーケンサ	189
シーケンス・コントローラ	167, 170, 178
時分割	101
時分割(多重化)バス	42, 138, 139, 143
終端	117, 142
シングルチップ・コンピュータ	10, 20, 35, 142, 167, 186
スイッチ	166, 170
スタティック駆動	120, 124
ステッピング・モータ	99, 113, 115
ストローブ	76
ストローブ信号	77, 78
スライド・スイッチ	94
すり合わせ接点	62, 94
スリーステート	43, 52, 53, 58
正帰還	82
絶縁	71
接点	47, 118
セットアップ時間	56, 112, 116
セマフォ・ビット	188
センサ	48, 83
セントロニクス型	79, 81, 113, 116
セントロニクス型インターフェース	140
セントロニクス型ハンドシェイク	115
ソリッド・ステート・リレー	118, 120

【タ行、ナ行】

ダイナミック駆動	101, 118, 123, 125
ダイナミック・スキャン	51, 68, 70, 174
多重化	143
チャタリング	17, 61, 62, 71, 127, 133
直接メモリ・アクセス	149
直流サーボ・モータ	183
直列通信回路	151

つき合わせ接点	62, 94
デジタル・アナログ変換器	97
デジタル・グラウンド	89, 137
デジタル・スイッチ	69, 170, 174
データ・ストロープ信号	97
データ・セットアップ時間	
.....	31, 32, 34, 104, 105
データ・バス	41
データ・ホールド時間	34, 35, 104, 136
デバッグ	19, 191, 194
デュアル・ポート RAM	187
テン・キー	66
電球	118
電磁カウンタ	113
電流ループ	74, 152
トグル・スイッチ	62, 94, 170, 174
トランスジューサ	48, 83
トランスバレント・ラッチ	35, 38
トランスバレント(透過型)・ラッチ	
.....	103, 105, 106, 138
トランスファ接点	65
ドループ率	86
ノートン・アンプ	74

【ハ 行】

パーソナル・コンピュータ	
.....	10, 70, 143, 161
バス・サイクル	
.....	24, 30, 31, 33, 56, 176
バス・スレーブ	24, 42, 54, 103
バス・マスタ	42, 43
バス・バッファ	26
発光ダイオード	97, 118, 120
パルス・エンコーダ	185, 188
パルス幅変調波	184
パルス・モータ	99
パワー MOS FET	119
パワー・リレー	131

ハンドシェイク

.....	76, 113, 115, 140, 178
ヒステリシス	60, 65, 82
ビデオ RAM	154
フォト・カブラ	
.....	71, 120, 123, 138, 178, 189
複数 CPU システム	161
符号化	51, 61, 67
ブラウン管	10
フリップフロップ	79
プリンタ	116, 140
ブルアップ	117
ブルアップ抵抗	72, 112, 113
フル・キー	68
フル・キーボード	51
フル・デコード	52, 91, 179
プログラマブル・カウンタ	99, 114
プロセス制御用	167
プロッタ	116, 140
分散処理	151, 161, 183
並列処理	151
ポーリング	49, 50, 76, 78
ホールド時間	112, 116
ポテンショメータ	83, 84, 182
ボリューム	83

【マ 行】

マイクロ・スイッチ	62, 63, 64, 94
マシン・サイクル	22
マルチプレクサ	59, 60, 72
モニタ・テレビ	154
モニタ・プログラム	192, 194

【ラ 行】

リード・リレー	129, 134
リニア・デコード	91, 93
リミット・スイッチ	61, 63, 184, 189
リレー	61, 71, 105, 108, 118,
.....	120, 128, 136, 166, 189

リレー効率	127
レベル検出	81
ローカル・エリア・ネットワーク	
.....	149, 151
ロータリ・スイッチ	
.....	51, 67, 69, 94, 170, 174

【ヤ 行】

有極リレー	129
優先順位	51, 67
誘導性負荷	127

【ワ 行】

ワイヤード OR	44
割込み	49, 50, 178

【欧 文】

A-D コンバータ	47
A-D 変換器	48, 61, 81, 85, 87
BBM	95
CRT	10, 154
CRT コントローラ	154, 158, 161
CRTC	154, 155
D フリップフロップ	
.....	25, 32, 101, 104, 110, 136, 138

D-A 変換	146
D-A 変換器	97, 100, 110, 134
DMA	50, 85, 100, 132, 151, 161
DPDT	94
Ethernet	149
FIFO	49, 78, 163, 187
GPIB	47, 79, 142, 149
IEEE-488	47, 142, 149
ITV カメラ	166
LAN	149, 151
LCD	118, 124
LED	72, 97, 118, 123, 166, 182
MBB	95
MOTEL	144, 145
PLL	161
PWM	184
R-2R ラダー	146
RS-232C	12, 47, 74, 138, 149, 170
SPST	94
SSR	118, 120
VRAM	154

著者略歴

あぜ つ あき ひろ
畔 津 明 仁

1954年 福岡県で生まれる

1977年 東京工業大学工学部卒業

1979年 東京工業大学大学院にて修士号を取得

1982年 同大学院博士課程を中退

現 在 電子機器の設計・開発に従事

実用インターフェース設計法

1985年3月30日 初版発行

©畔津明仁 1985

1989年8月1日 ■ 7版発行

著 者 畔 津 明 仁

発行人 神 戸 一 夫

発行所 CQ出版株式会社

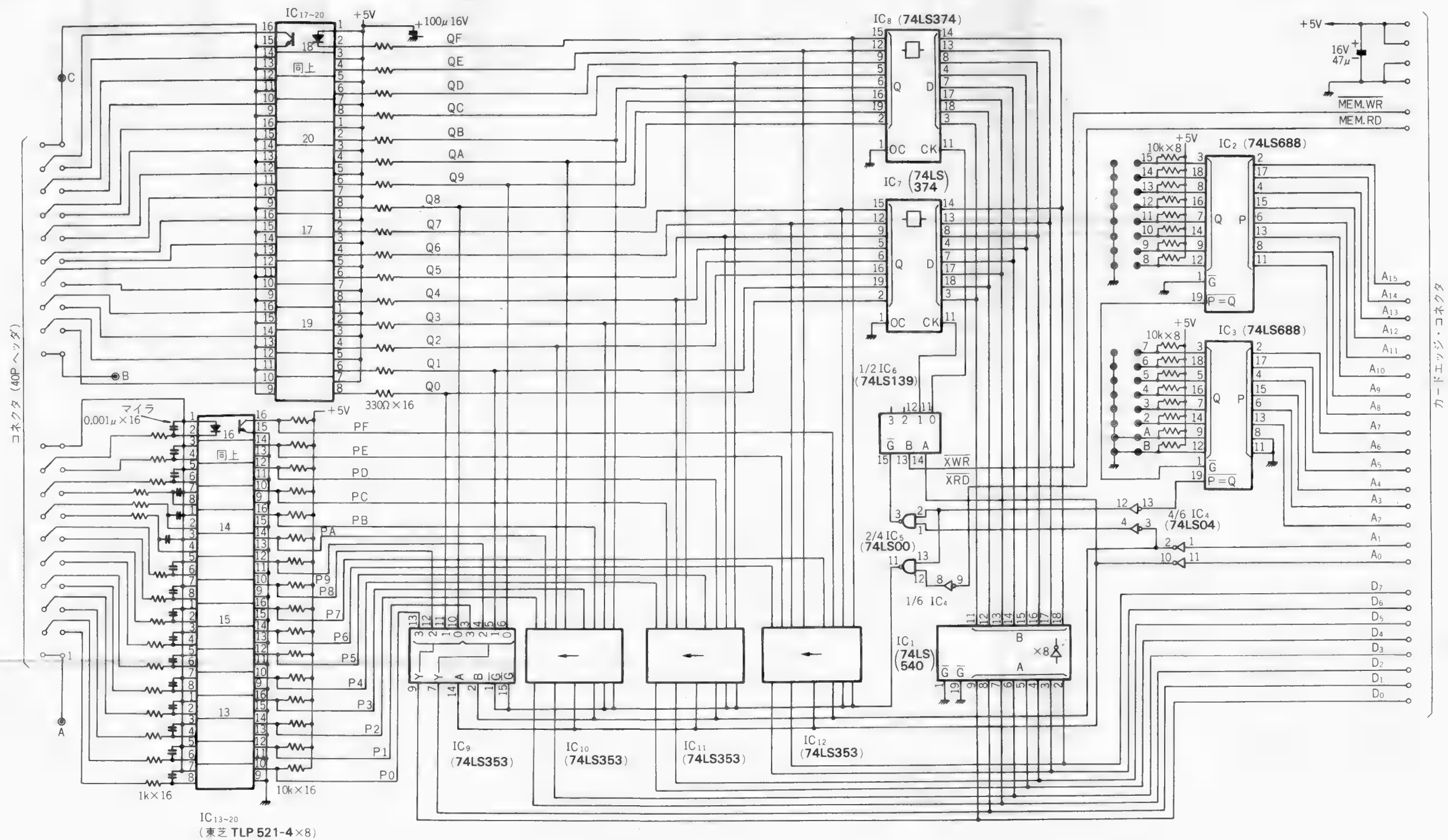
東京都豊島区奥鴨1-14-2 (〒170)

電話 03 (947) 6311 (代) 振替東京0-10665

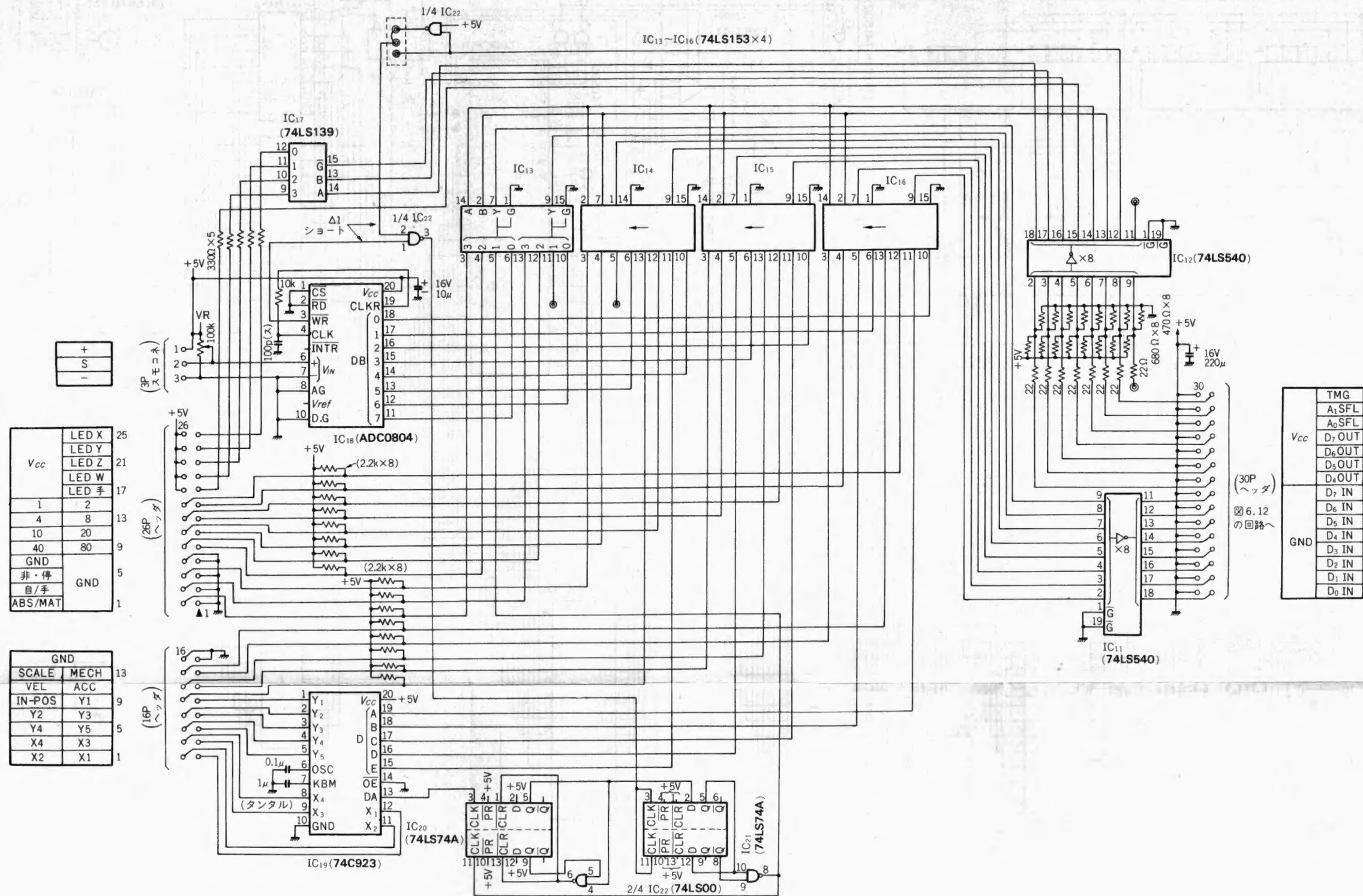
(定価はカバーに表示してあります)

乱丁、落丁本はお取り替えます

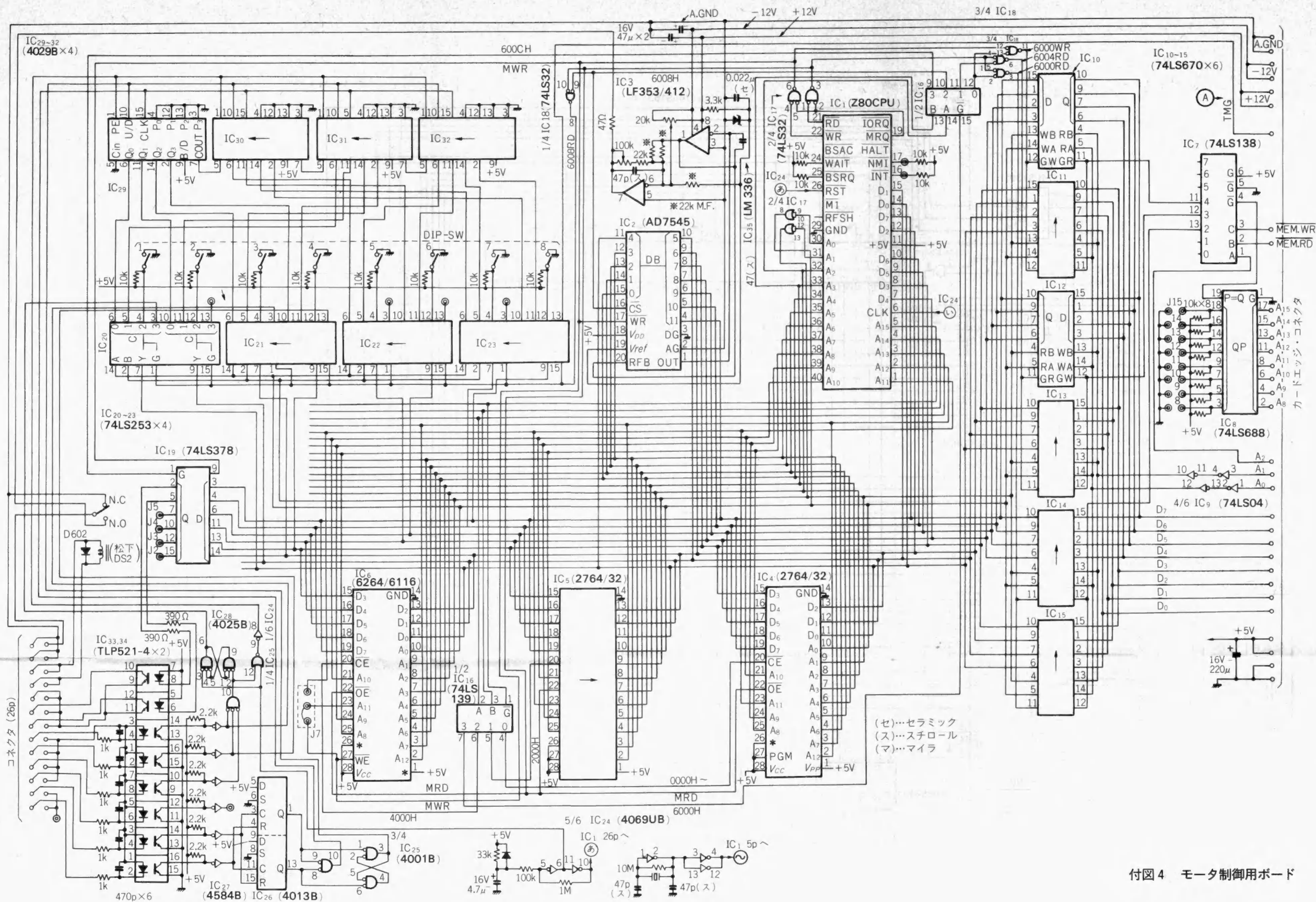
制作・写植 都写植／印刷・製本 啓文堂



付図2 フォト・カブラで絶縁された入出力インターフェース・ボード



付図3 操作箱用インターフェース・ボード





CQ出版社 定価1,440円(本体1,398円)
ISBN4-7898-3200-7 C3055 P1440E